HARNESSING RESILIENCE: BIASED VOLTAGE OVERSCALING FOR PROBABILISTIC SIGNAL PROCESSING

A Thesis
Presented to
The Academic Faculty

by

Jason George

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2011

HARNESSING RESILIENCE: BIASED VOLTAGE OVERSCALING FOR PROBABILISTIC SIGNAL PROCESSING

Approved by:

Paul Hasler, Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

David Anderson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Vincent J. Mooney III
School of Electrical and Computer Engineering
Georgia Institute of Technology

Bonnie Heck Ferri
School of Electrical and Computer Engineering
Georgia Institute of Technology

Thomas Conte
School of Computer Science
Georgia Institute of Technology

Date Approved: 22 August 2011

To my parents,

Mike and Gisela George,

who did all the things over the years so my brother and I might have opportunities such as this one.

ACKNOWLEDGEMENTS

First, and foremost, I would like to thank my doctoral advisor David Anderson, who was kind enough to take me in when my research was so far from what the rest of his group was doing. Along the way David has consistently offered guidance and provided opportunities that not only made the completion of my dissertation possible, but helped launch my career in a direction I had only hoped might be an option. Without question, David is one of the kindest and most generous people I have even known and a graduate student would be lucky to have such a wonderful advisor.

Second, I would like to thank Krishna Palem, who advised me for quite some time. It was Krishna who provided me with such an unorthodox and interesting dissertation topic. Had it not been for an opportunity that took Krishna away from Georgia Tech, which he generously extended to me, I would have completed my dissertation with Krisha. Even so, Krisha continued to offer both research and career advice from his new post, for which I am genuinely grateful.

In the same vein, I want to thank my dissertation committee—Paul Hasler, Vincent J. Mooney III, Bonnie Heck Ferri, and Thomas Conte—who were nice enough to take time away from their insanely busy schedules to help me complete my dissertation. In particular, both Paul and Vincent provided invaluable insight and expertise in circuit design that were instrumental in the completion of my work.

Many of my fellow graduate students also deserve mention for their contributions to the completion of my doctoral degree—far more than I could enumerate here. My colleges and predecessors from the CREST group laid the foundation for my topic and helped immensely with my work. Lakshmi Chakrapani, Bo Marr, Bilge Akgul, and Pinar Korkmaz all contributed significantly and both Lakshmi and Bo continue to offer advice long after their graduations. Similarly, my colleagues from the CADSP group have helped in many

aspects of my dissertation work. In particular, Brian Degnan has consistently (and generously) helped me sort out every Cadence issue that halted my progress and whenever a signal processing subtlety was eluding me Brian Gestner, Ken Chiu, and Devangi Parikh were always happy to help.

Finally, it was my parents who set me on the path that eventually placed me here. They always made sure I had the tools I needed to succeed and provided the guidance and encouragement—regardless of how big or small the task—that helped see me through whatever endeavor was at hand. Without my parents support, it is unlikely I would have ever been able to accomplish all the things that helped me get here and I could never thank them enough.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A central component of modern computing is the idea that computation requires determinism. Contrary to this belief, the primary contribution of this work shows that useful computation can be accomplished in an error-prone fashion. Focusing on low-power computing and the increasing push toward energy conservation, the work seeks to sacrifice accuracy in exchange for energy savings.

Probabilistic computing forms the basis for this error-prone computation by diverging from the requirement of determinism and allowing for randomness within computing. Implemented as probabilistic CMOS (PCMOS), the approach realizes enormous energy savings in applications that require probability at an algorithmic level. Extending probabilistic computing to applications that are inherently deterministic, the biased voltage overscaling (BIVOS) technique presented here constrains the randomness introduced through PCMOS. Doing so, BIVOS is able to limit the magnitude of any resulting deviations and realizes energy savings with minimal impact to application quality.

Implemented for a ripple-carry adder, array multiplier, and finite-impulse-response (FIR) filter; a BIVOS solution substantially reduces energy consumption and does so with improved error rates compared to an energy equivalent reduced-precision solution. When applied to H.264 video decoding, a BIVOS solution is able to achieve a 33.9% reduction in energy consumption while maintaining a peak-signal-to-noise ratio of $35.0dB$ (compared to $14.3dB$ for a comparable reduced-precision solution).

While the work presented here focuses on a specific technology, the technique realized through BIVOS has far broader implications. It is the departure from the conventional mindset that useful computation requires determinism that represents the primary innovation of this work. With applicability to emerging and yet to be discovered technologies, BIVOS has the potential to contribute to computing in a variety of fashions.

CHAPTER I

INTRODUCTION

1.1 A Historical Perspective: Power Consumption and Computing

More than ever before, energy is a topic that is capturing global attention. Both governments and private investors are launching projects intended to address rising global energy demands while reducing the environmental footprint of energy production (the U.S. alone dedicated $16.8 billion to energy recovery in 2009 [79]). In conjunction, consumers and manufactures are actively pursuing energy conservation in an effort to combat the rising costs associated with increased energy demand. This has lead to the emergence of products ranging from automobiles to light bulbs that are far more efficient than their predecessors. While this general trend toward energy efficiency is relatively new, the computing domain has long considered energy consumption issues by necessity. Specifically, applications requiring mobility required batteries for operation and the use of batteries dictated a limited power supply. Because of this limited power supply, it was important to use what power was available judiciously in order to maximize battery life.

For several decades addressing mobility concerns has largely been the sole purpose of low-power computing. Systems that allowed for "on-the-grid", or fixed, computing had a limitless power source from a wall outlet and there was little need for power conservation. Beyond the rare instances where high power consumption levels created heat dissipation issues, performance came first and energy efficiency received little attention if battery life was not a concern. Quite the opposite in the mobile domain, increased performance was primarily achieved through improvements in efficiency. As mobility demands continued to increase, energy conservation was the only option to meet rising performance requirements while maintaining battery life. As a result, mobility has been the primary force behind recent advancements in low-power computing.

In the distant past, however, power consumption was a very real concern for computing

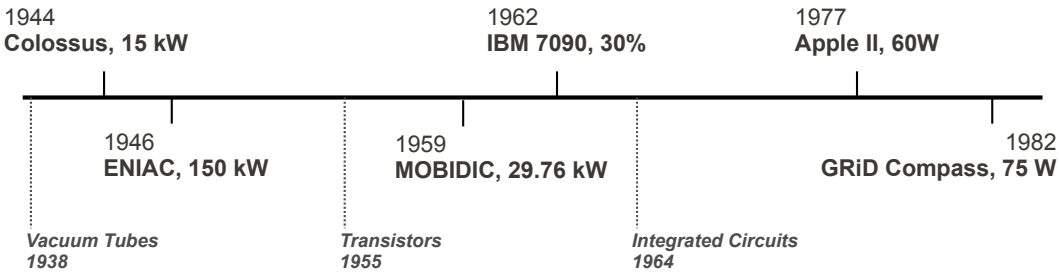


Figure 1: Time line highlighting the progression of power consumption in computing. The earliest electronic computers, such as the Colossus and ENIAC, consumed tremendous amounts of power due to their vacuum-tube designs. With the invention of the transistor, power consumption was reduced substantially in the next generation of machines (the IBM 7090 used 30% of the power of the vacuum-tube computer it replaced). Finally, integrated circuits further reduced power consumption making the personal computer and mobile computing possible.

in general. The earliest electronic computers utilized vacuum tubes to perform calculations [91]. With a design similar to an incandescent light bulb, vacuum tubes required large temperature differentials for proper operation. These large temperature differentials consumed substantial amounts of power that was lost as waste heat and created mechanical stress on internal components during heating and cooling cycles (at power-up and power-down respectively) [95]. Because of this repeated stress, vacuum tubes had high failure rates and where most likely to fail during power cycles [89]. As a result, many vacuum tube based computers where never turned off once they had been turned on—operating 24 hours a day, 365 day a year.

One of the earliest vacuum-tube computers was the Colossus [88, 91]. Completed in 1944 as part of the English war effort during World War II, the Colossus was designed to break German messages that had been encrypted using the Lorenz cipher machine. It was the first digital electronic computer and the final design of the machine utilized $2,400$ vacuum tubes consuming $15kW$ of power. (333 times the $45W$ supplied to a Macbook Air [2]). Following the Colossus, the ENIAC was completed in 1946 for the United States Army. It was designed to calculate artillery firing tables and was the first general-purpose electronic computer [91, 89]. Employing $17,468$ vacuum tubes, the machine consumed an astounding $150kW$ of power and was in continuous operation from July 29, 1947 through October 2,

1955 (consuming a total of $10.8TWh$ of power). By comparison, an average home in the United States consumes $11,040kWh$ annually and the operation of the ENIAC could have powered just under $1,000$ modern homes for an entire year [80].

With the invention of the transistor, transistorized computers began replacing vacuum tube designs from 1955 onward [91]. A transistor computer utilized circuit boards full of individual transistors wired to form computing circuits. Despite the fact that early transistors were even less reliable than vacuum tubes, they began replacing vacuum tubes because of their smaller size and the fact that they consumed far less power. With the invention of silicon junction transistors, transistors had an indefinite service life and became substantially more reliable than vacuum tubes. This afforded transistorized computers a benefit in size and cost (both initial and operating) when compared to their vacuum tube counterparts.

The MOBIDIC, short for "MOBile Digital Computer", was one of the earlier transistorized computers. The machine was designed to automate the routing of battlefield data for the U.S. Army and mounted in a semi trailer for mobility. A second trailer carrying a generator set supplied the $29.76kW$ of power the machine required [93]. In contrast, one of the earliest commercial transistorized computers was the IBM 7090. It was designed for scientific computing as a transistorized version of the IBM 709 vacuum-tube, mainframe computer. Compared to the 709 vacuum-tube design, the 7090 consumed only 30% of the power with a 6 fold increase in computing power [32, 92].

The integrated circuit followed on the heels of the invention of the transistor and further improved on the power, size, and cost benefits offered by transistors. So much so, that the invention of the integrated circuit led to the personal computer. One of the earliest, the Apple II, consumed a mere $60W$ of power [1]. With such meager power requirements, truly mobile computing was suddenly a possibility. The GRiD Compass, released in 1982 as the first laptop computer, signaled the dawn of mobile computing. Requiring only $75W$ of power, the machine ushered in a new age of low-power computing targeted specifically at mobility [90]. Because mobility limited power resources to what could be carried, prudent energy use was paramount to conserve what power was available and to extend the

operational life as long as possible.

Since the commercialization of the integrated circuit, not a lot has changed in terms of power consumption and computing. While there are some exceptions, low-power computing has largely been reserved for the mobile domain. Systems with access to the power grid have enjoyed a virtually limitless energy supply and the *realtively* low energy requirements of modern computers has diminished the importance of power consumption. As a result, there has been little need to conserve energy for "on-the-grid" computing.

1.2 Global Energy Consumption and the Impact of Computing

Beyond the domain of computing, energy consumption in general has traditionally garnered little attention. Evident of the fact that energy use has primarily been an afterthought for decades, per capita energy consumption within the United States has increased by 37% over the period of 1960 to 2007 [77]. Total energy consumption for the same time period is up 125% within the U.S. [81]. Globally the trends are similar with a per capita increase of 35% from 1971 to 2007. Generally speaking, energy use has been ignored in pursuit of ever increasing convenience, functionality, and performance.

To some extent, however, this philosophy of performance above all else is changing. Global energy consumption is growing dramatically, driven largely by the rapid industrialization of China and India [78], pushing energy prices increasingly higher. Coupled with this, increasing global temperatures have raised significant concerns about the impact of carbon emissions on global climate change [22, 83] (roughly 70% of world wide electricity production in 2004 relied on fossil fuels and that number is projected to grow to 90% by 2030 [78]). In contrast to decades past, these rising economic and environmental concerns have lead to a recent movement toward energy conservation.

With global energy consumption in mind, computing might seem like a strange place to look for energy savings. The domain, however, (driven by consumer electronic devices) accounts for approximately 11% of household electricity consumption in the United States [70]. This translates to 4% of annual U.S. electricity consumption, or 1.6% of annual energy

consumption. Assuming roughly half of computing energy consumption is derived from residential use and the other half from commercial use, as shown in [23], computing resources can be credited with 8% of total U.S. electricity consumption. In a study commissioned by the Swiss Centre for Technology Assessment [38], Germany's computing resources are cited as contributing an estimated 7.1% (or $38TWh$) to national electricity consumption in 2001 [23]. The study further cites Swiss computing resources as accounting for 3.6% ($1.8TWh$) of national electricity consumption at the time of writing and goes on to project that the introduction of pervasive computing could drive that number as high as 7% ($6TWh$) by 2012.

Of the total electricity consumption within the computing domain, server installations represent a substantial portion. This is driven largely by the enormous growth of the Internet (41% annually over the last 15 years [14] with over 700 million addressable devices as of 2009 [15]). In a symbiotic cycle, the growth of the Internet has provided increased services, which has lead to an increase in Internet users driving the demand for even more services. Fulfilling this demand, data centers containing hundreds (to thousands) of servers consume enormous amounts of electricity. They require energy for operation of the servers and even more for the associated cooling necessary due to the tremendous amount of waste heat the servers generate:

> Total power used by servers represented about 0.6% of total U.S. electricity consumption in 2005. When cooling and auxiliary infrastructure are included, that number grows to 1.2%, an amount comparable to that for color televisions. The total power demand in 2005 (including associated infrastructure) is equivalent (in capacity terms) to about five 1000 MW power plants for the U.S. and 14 such plants for the world. The total electricity bill for operating those servers and associated infrastructure in 2005 was about $2.7 B and $7.2 B for the U.S. and the world, respectively [39].

Beyond the server space, smaller systems are beginning suffer from heat dissipation issues as well. For years the personal computer industry measured performance in megahertz and Intel, among others, marketed their products with a line of ever increasing clock rates. As transistor densities increased and clock rates climbed, the industry eventually reached a point where heat dissipation was such a problem that it became a barrier to increased

performance. So much so that according to Linley Gwennap, improvements in the central-processing units (CPUs) are no longer driving computing performance:

> Today's CPUs have megahertz to burn but are throttled by the amount of heat that the system can pull out. Reduce the CPU power by 10% and you can push the clock speed up to compensate, turning power into performance gains. Most CPU design teams are now more focused on the power budget than on the timing budget [28].

This "power wall" that Gwennap suggests is a primary factor limiting processor performance. A 2008 exascale study targeted at achieving ExaFLOP computation (10^{18} floating-point operations per second) cites power concerns as the force behind the wall that the clock rates have hit, remaining flat since the early 2000s [37]. Further, the challenges posed by the power wall are only projected to grow. The same study goes on to cite power consumption as "the single most difficult and pervasive challenge" to reaching the ExaFLOP goal [37].

1.3 Achieving Low Power Computation

Driven largely by the push for ever increasing performance in the mobile domain, low-power computing has received significant attention. A central component to this quest to reduce energy consumption is the CMOS power consumption equation (Equation 1) [55]. The equation splits the power consumption of CMOS transistors into three components: dynamic power consumption, short-circuit power consumption, and leakage power consumption. Dynamic power consumption ($ACV_{dd}^2 f$) is the power that is consumed by switching transistors on and off through normal circuit operation. It is determined by a combination of circuit activity A, line capacitance C, supply voltage V_{dd}, and operating frequency f. Short-circuit power consumption ($\tau A V_{dd} I_{short} f$) occurs when a gate output changes and the pull-up and pull-down networks are both momentarily on during the change. The resulting short circuit between the supply and ground rails allows current to flow from power to ground. Short-circuit power consumption is defined by the amount of time the short circuit lasts τ (a function of supply voltage), circuit activity, supply voltage, short-circuit current I_{short} (also a function of supply voltage), and operating frequency. Finally, leakage power consumption ($V_{dd} I_{leak}$) is caused by current that constantly flows, or leaks, through gates when powered.

It is a combination of supply voltage and leakage current I_{leak} (a function of transistor threshold, or switching, voltage).

$$P = ACV_{dd}^2 f + \tau A V_{dd} I_{short} f + V_{dd} I_{leak} \quad (1)$$

From Equation 1, it is apparent that reductions in supply voltage yield substantial reductions in power consumption. A linear decrease in operating voltage results in a quadratic reduction in dynamic power consumption with the added bonus of linear reductions in short-circuit and leakage power consumption. Because of this quadratic relationship between supply voltage and power consumption, there has been substantial research in voltage reduction techniques.

Of the various techniques proposed, the most effective approach to reduce power consumption is to eliminate supply voltage altogther. Clock gating is one example of these "power-down" techniques where the clock signal is turned off for unused portions of a circuit. As a clock tree consumes up to 30% of a processors power [55], this can lead to substantial savings. Another is partial memory shutdown where data is remapped and caches intercept memory accesses to minimize memory usage [66, 69]. As the memory system and logic buses are also significant sources of power consumption, this can also lead to large energy savings [55].

Short of completely turning unused portions of the circuit off, then next best alternative to reduce power consumption is to reduce supply voltages for operational portions of the circuit—the extreme case of subthreshold voltage scaling reduces supply voltages below transistor threshold levels [11, 12, 13, 85]. Unfortunately, however, reductions in supply voltage (above the threshold volatage) also reduce the maximum operating frequency of the circuit and impact overall performance (Equation 2) [55]. To combat this reduction in performance, dynamic voltage scaling continually adjusts supply voltages to dynamically meet changing timing requirements [51]. Parallel processing extends the technique by dividing workloads across multiple processors, allowing each processor to operate at a lower voltage while maintaining timing requirements [17]. In a similar fashion, pipelining subdivides individual instructions and allows a single processor to begin processing new instructions

before previous instructions have completely executed. As a result, pipelining also allows a processor to operate at a lower voltage by artificially increasing the completion rate (or operating frequency) of a circuit [55].

$$f_{max} \propto (V_{dd} - V_{th})^2 / V_{dd} \tag{2}$$

Beyond the performance penalty that comes with voltage scaling, reducing the supply voltage, V_{dd}, necessitates a reduction in threshold voltage, V_{th}, to maintain proper circuit switching. Given the relationship between leakage current and V_{th} (shown in Equation 3) a reduction in threshold voltage leads to an increase in leakage power consumption [55]. As a result, reductions in V_{th} to maintain circuit performance in a reduced-voltage configuration can quickly make I_{leak} a significant power contributor. Further complicating the issue, the constant march of Moore's law to reduce feature sizes actually requires reductions in supply voltage and amplifies static power consumption [35].

$$I_{leak} \propto exp(-qV_{th}/\kappa T) \tag{3}$$

Several techniques attempt to address the increase in leakage current due to threshold-voltage scaling. Multithreshold approaches employ the use of high V_{th} transistors to provide a virtual power rail to low V_{th} transistors. The virtual power rail then allows transistors to be put to sleep with virtually no leakage current while still permitting the use of low V_{th} transistors to improve circuit performance [53, 56]. Stacked transistor designs accomplish the same by adding a second low V_{th} transistor, creating stacked transistor pairs that reduce the leakage current in "off" devices [29, 33, 57]. Combining the two approaches, the sleepy stack techniques utilize a high V_{th} transistor for forced stacking along with a high V_{th} sleep transistor in parallel to provide a constant current source when powered [65]

Among the more unconventional approaches to reducing power consumption, schemes employing multiple voltage sources attempt meet timing requirements by applying a high supply voltage to gates along the critical path of a circuit while reducing the supply voltage along non-critical-path elements [18, 51, 82, 100, 101]. Others attempt to reduce supply voltages beyond timing requirements, allowing critical path errors to occur and correcting any

that do. These include efforts ranging from a collection of software-based, signal-processing techniques designed to detect and correct errors in soft DSP [30] to a fully hardware-based solution that aggressively scales voltages until errors are detected in Razor [3].

Extending the practice of error prone computing, probabilistic CMOS (PCMOS) also attempts to scale supply voltages beyond tolerable limits. Unlike other techniques, however, PCMOS performs no error correction to maintain deterministic operation. Instead PCMOS attempts to characterize the probability associated with any errors and relies on applications to either capitalize on, or tolerate, the resulting randomness. In this way, PCMOS is able to achieve power savings through voltage overscaling without the added overhead of error correction circuitry [60, 62].

CHAPTER II

PROBABILISTIC COMPUTING

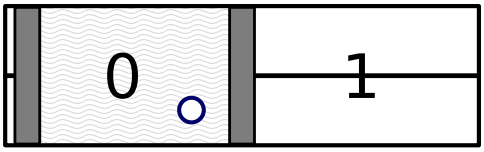
2.1 Fundamentals of Probabilistic Computing

Fundamentally, computing is a mathematical practice. It is a process of determining facts through calculation. As such, there is a sense of definitive truth in the practice of computing. It is this definitiveness, or determinism, that forms the basis of the modern digital computer. So much so, that it is implicitly expected that any calculation performed by a computer is absolutely correct.

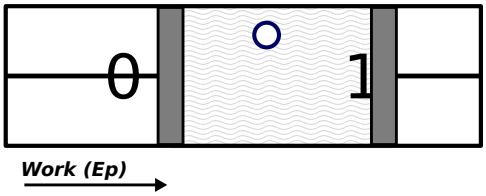
This expectation of infallibility in computing is underscored by the Pentium floating-point division bug (first publicized by Thomas Nicely in October of 1994). The bug was caused by an omission of five entires in a lookup table used for floating-point division. As a result of these five missing entries, the Pentium processor would occasionally calculate floating-point divisions incorrectly (estimated at one error every 27,000 years for a typical user [74]) with errors limited to a maximum magnitude of 2^{-14} [67]. Despite the extremely unlikely potential for error, and the extremely small deviations, there was a public outcry once the bug came to light resulting in Intel adopting a replacement policy that ultimately cost the company $475 million [94].

The crux of modern, digital computation is that the underlying hardware always returns a correct result. A NOT function always returns $X = \overline{Y}$ and an AND function always returns $X = A \cdot B$. The entire system is predicated on this level of determinism at the lowest level and the design of subsequent layers of complexity (architecture, software, etc.) rely on it. Computing to this level of accuracy obviously incurs a design cost. Computing systems are extremely complex and validating functionality is expensive. Additionally, however, there is a computational cost of maintaining such a high level of accuracy that is often overlooked.

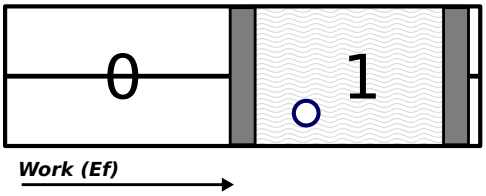
Example 1: As a thought experiment, consider a theoretical switch made of a cylinder containing a single gas molecule. A binary zero is represented by the gas molecule's presence on the left half of the cylinder; a binary one is conversely represented by the molecule's presence on the right half of the cylinder. A pair of pistons, one on each side of the cylinder, set the switch's state by constraining the gas molecule to one half of the cylinder or the other. Between the pistons the gas molecule is free to move within the cylinder and thermal excitation causes random drift.



Assuming the theoretical switch is initially at state zero, switching to state one requires repositioning the pistons to force the gas molecule to the right half of the cylinder. Partially switching the pistons requires a fixed amount of energy E_p. Because the gas molecule is free to move between the pistons, leaving the cylinder in this state allows for a fixed probability that the gas molecule will drift to the left half of the cylinder resulting in an error.



To guarantee error free operation the pistons must be fully switched to ensure the gas molecule is unable to reach state zero. This requires a fixed amount of energy E_f and, because a full switching always requires more movement than a partial switching, $E_f > E_p$. It is the difference between E_f and E_p that represents the computational cost of deterministic computing.



□

Based on the idea that there is a minimal energy cost of computing, the fundamental lower limit was calculated to be $\kappa T ln(2)$ [42, 52]. In the context of the modern computer [52], it was determined as the minimal energy required to switch the state of a single electron. Predicted by von Neumann [84], this limit was based on the notion of deterministic computing. In truth, the physical behavior of objects, including modern microprocessors, is best defined statistically. The illusion of determinism arises from tremendous expenditures of energy to ensure the probability of failure is virtually nonexistent.

Revisiting Example 1, allowing for partial, or lazy, switching reduces the energy requirements of the computation. The trade-off is that there is potential for the theoretical molecule drift to the left half of the cylinder resulting in an error. As a result, there is a fixed probability of a correct calculation and a fixed probability of an error. Defining this sort of partial switching as a probabilistic computation, probabilistic computing allows for energy savings by relaxing the accuracy requirements of computation.

Probabilistic computing then defines the probability of a correct operation as p where $0.5 \leq p \leq 1$. In turn, the probability of an error is defined as $1-p$. With this as a basis, the minimal energy cost of computing in a probabilistic sense is derived to be $\kappa T ln(2p)$ [61, 62]. Compared to a corresponding minimal deterministic operation, probabilistic computing theoretically saves $\kappa T ln(\frac{1}{p})$ energy at the expense of a loss in accuracy.

2.2 Probabilistic Arithmetic

With probabilistic computing as a basis, probabilistic arithmetic is defined such that operations have a probability of correctness p_i associated with each i^{th} bit position. As such, an n-bit probabilistic arithmetic operation is a function $\mathcal{O}_P$ that transforms two n-bit inputs (X and Y) into an m-bit solution (Z) where bitwise operations are calculated with an accuracy defined by each p_i. More formally, $\mathcal{O}_P$ is a function where $\mathcal{O} : X \in \{0,1\}^n \times Y \in \{0,1\}^n \rightarrow Z \in \{0,1\}^m$ and the probability of correctness is defined as $P =< p_0, p_1, \cdots, p_{m-1} >: 0 \leq p_i \leq 1$. The conventional (deterministic) function corresponds to the case where $P \equiv < 1 >$ [27]. In all other cases, a probabilistic arithmetic operation yields a solution that deviates from a corresponding deterministic operation $\mathcal{O}_D$

as defined by P.

The accuracy of a probabilistic arithmetic operation can then be measured as the resulting deviation from the corresponding deterministic operation. For a single trial of $\mathcal{O}_P$, the difference between $\mathcal{O}_P$ and $\mathcal{O}_D$ ($Z_P - Z_D$) represents the deviation introduced by probabilistic computing. Equivalently, $Z_P - Z_D$ is equal to the sum of any bitwise errors occurring in $\mathcal{O}_P$ weighted by the bit position of occurrence. Subdividing $\mathcal{O}_P$ into h bitwise operations then allows $Z_P - Z_D$ to be determined by summing the impact of any individual error at each j^{th} operation q_j. Defining an error vector $E =< e_0, e_1, \cdots, e_{h-1} >$ such that an error at operation j is indicated by $e_j \in \{0, 1\}$ (where 1 represents a bitwise error and 0 represents correct operation), a direction vector $D =< d_0, d_1, \cdots, d_{h-1} >$ where the direction of an error is indicated by $d_j \in \{-1, 1\}$ (a $0 \rightarrow 1$ bit flip equals 1 and a $1 \rightarrow 0$ equals -1), and a magnitude vector $M =< m_0, m_1, \cdots, m_{h-1} >$ such that the magnitude of an error is indicated by m_j (equal to 2^i as determined by the bit position of j); then allows the deviation introduced by $\mathcal{O}_P$ to be calculated in terms of individual bit errors. Given an error vector E, Equation 4 then represents the magnitude of the deviation introduced by $\mathcal{O}_P$.

$$\mathbf{MAG}_E = \sum_{j=0}^{j=h-1} d_j \cdot e_j \cdot m_j \tag{4}$$

Example 2: Consider a one-bit addition operation consisting of two bitwise operations (q_0 calculates the sum bit as $x_0 \oplus y_0$ and q_1 calculates the carry bit as $x_0 \cdot y_0$). Since the sum bit applies to bit zero and the carry bit applies to bit one, $m_0 = 2^0$ and $m_1 = 2^1$ implying $M =< 1, 2 >$. If $X = 1$ (0_b1) and $Y = 1$ (0_b1), then a deterministic addition operation $\mathcal{O}_D$ would yield $q_0 = 0$ and $q_1 = 1$ resulting in $Z_D = 2$ (0_b10). Implemented as a probabilistic addition operation $\mathcal{O}_P$, a single error at q_0 would cause a $0 \rightarrow 1$ bit flip resulting in $q_0 = 1$ with $e_0 = 1$ and $d_0 = 1$ ($E =< 1, 0 >$ and $D =< 1, 1 >$). Equation 4 then evaluates to 1 ($d_0 \cdot e_0 \cdot m_0 = 1 \cdot 1 \cdot 1 = 1$ for $j = 0$ and $d_1 \cdot e_1 \cdot m_1 = 1 \cdot 0 \cdot 2 = 0$ for $j = 1$). Verifying the calculation, $Z_P = 3$ (0_b11) and $Z_P - Z_D = 1$. □

Repeated over multiple trials, the individual bit errors introduced by $\mathcal{O}_P$ will vary due

to the randomness inherent in probabilistic computing. The probability of an error combination occurring is determined by the joint probability of the individual bitwise operations. As P is defined by bit position, the probability of correctness p_j for the j^{th} operation is then determined by the bit position of q_j. In turn, the probability of an error at q_j is defined as $1 - p_j$. The resulting probability of a particular error combination E, or deviation $(Z_P - Z_D)$, is then defined in Equation 5.

$$\mathbf{P}_E = \prod_{j=0}^{j=h-1} \Big(e_j \cdot (1 - p_j) \Big) + (\bar{e}_j \cdot p_j) \tag{5}$$

Example 3: As in Example 2, assume a one-bit addition operation with a single bit error at q_0 ($E =< 1, 0 >$). If $P \equiv< 0.9 >$ then $p_i = 0.9$ for all bit positions i and as a result $p_j = 0.9$ for all bitwise operations. Equation 4 then evaluates to 0.1 for q_0 $\big(\big(e_0 \cdot (1 - p_0)\big) + (\bar{e}_0 \cdot p_0) = \big(1 \cdot (1 - 0.9)\big) + (0 \cdot 0.9) = 0.1\big)$ and 0.9 for q_1 $\big(\big(e_1 \cdot (1 - p_1)\big) + (\bar{e}_1 \cdot p_1) = \big(0 \cdot (1 - 0.9)\big) + (1 \cdot 0.9) = 0.9\big)$ yielding $\mathbf{P}_E = 0.09$. □

Given the deviation due to a particular error combination (Equation 4), the probability of a particular error combination occurring (Equation 5), and 2^h possible combinations of h bitwise errors, the expected deviation of a probabilistic operation $\mathcal{O}_P$ can be determined as defined in Equation 6 by summing over each k_{th} error vector $E(k)$.

$$\mu(\mathcal{O}_P - \mathcal{O}_D) = \sum_{k=0}^{k=2^h-1} \mathbf{MAG}_{E(k)} \cdot \mathbf{P}_{E(k)} \tag{6}$$

Similarly, the expected mean-squared error for a probabilistic operation $\mathcal{O}_P$ can be calculated as the summation of the squared deviations weighted by error-combination probability (Equation 7).

$$\mathbf{MSE}(\mathcal{O}_\mathcal{P}) = \sum_{k=0}^{k=2^h-1} \mathbf{MAG}^2_{E(k)} \cdot \mathbf{P}_{E(k)} \tag{7}$$

Example 4: Continuing with Example 3, a one-bit addition operation with two bitwise operations has four possible error combinations ($E(0) =< 0, 0 >$, $E(1) =< 1, 0 >$, $E(2) =< 0, 1 >$, and $E(3) =< 1, 1 >$) resulting in four possible error magnitudes ($M_{E(0)} = 0$, $M_{E(1)} = 1$, $M_{E(2)} = 2$, and $M_{E(3)} = 3$). If $P \equiv< 0.9 >$ then the probability of each of the four error combinations occurring is equal to $P_{E(0)} = 0.81$, $P_{E(1)} = 0.09$, $P_{E(2)} = 0.09$, and $P_{E(3)} = 0.01$. From Equations 6 and 7 respectively, the expected deviation $\mu(\mathcal{O}_P - \mathcal{O}_D) = 0.3$ and the expected mean-squared error $\mathbf{MSE}(\mathcal{O}_\mathcal{P}) = 0.54$. □

2.3 Probabilistic CMOS

The foundations of digital logic, on which CMOS computing is built, center on noise immunity and error-free operation. By definition, digital logic groups analog signals into discrete voltage bands in order to reduce susceptibility to voltage fluctuations. Binary logic allows for just two bands, a zero or a one, maximizing circuit noise immunity by allowing for a noise margin with voltage swings up to half of the operating voltage ($V_{dd}/2$).

As is the case for probabilistic computing in general, Probabilistic CMOS (PCMOS) attempts to relax these rigorous accuracy constraints allowing standard CMOS to be subjected to noise interference resulting in bit errors. While any noise source could potentially be used as a source of randomness to render CMOS probabilistic, thermal noise is considered due to projections that it will become an impediment to future technology scaling [36, 58, 75]. In the case of thermal noise, thermal excitation causes electrons to perpetually change velocities resulting in voltage fluctuations. Should a fluctuation cause signal voltage to exceed the digital noise margin, $V_{dd}/2$, the result is a bit error (Figure 2).

Reductions in supply voltage, in turn, decrease the noise margin for a circuit. This both reduces the circuit power consumption and increases susceptibility to thermal voltage fluctuations. Scaling supply voltages beyond the point where noise margins can contain thermal fluctuations results in bit errors. In this way, PCMOS attempts to trade accuracy for energy savings.

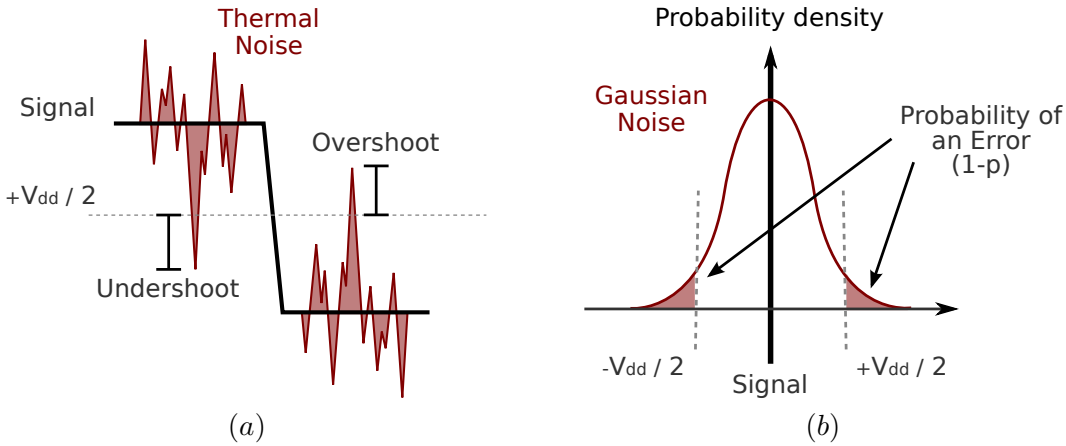


Figure 2: Thermal noise interference in digital voltage signals: (a) Thermal noise is an additive noise source that causes fluctuations in the intended voltage of a signal. If noise causes the operating voltage of a signal at V_{dd} to drop below $V_{dd}/2$ the resulting undershoot causes a bit flip ($1 \rightarrow 0$). Correspondingly, if noise causes the operating voltage of a signal at Gnd to rise above $V_{dd}/2$ the resulting overshoot also causes a bit flip ($0 \rightarrow 1$). (b) Thermal noise can be represented as a probability density function (PDF). Since thermal noise is additive, zero-mean, and Gaussian, the resulting PDF is centered at the operating voltage of the affected signal. The portion of the PDF that extends beyond $\pm V_{dd}/2$ represents the probability of a signal error occurring due to a bit flip.

2.4 Probabilistic Gates

As noise is applied to PCMOS gates it is considered to be a random process and is characterized by the resulting statistical interference on gate signals. In the case of thermal noise, the interference is characterized by a zero-mean, Gaussian distribution that is additive [40, 76]. The result is a fluctuation in signal-voltage levels that follows a probability distribution defined by thermal noise and centered at the intended voltage level (Figure 3). The portion of the PDF that extends beyond the switching point of the gate represents the probability that thermal noise will cause a voltage fluctuation resulting in a bit flip. Conversely, the probability of correctness is represented by the portion of the PDF that does not violate the switching point.

As a result, given an operating voltage and noise RMS, PCMOS gates can be characterized for energy consumption and probability of correctness. In turn, varying operating voltage at a fixed-noise RMS yields an energy-probability (Ep) profile. Of particular interest, the quadratic reduction in energy consumption due to voltage scaling combined with a linear decrease in accuracy due to thermal noise yields a quadratic relationship between

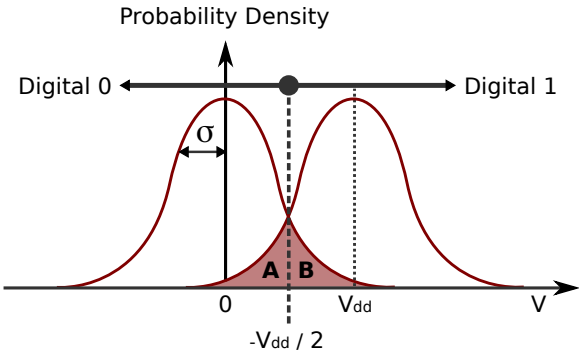


Figure 3: Represented as a probability-density function (PDF), thermal noise applied to an inverter causes statistical fluctuations centered at the input voltage. If the inverter input is at a 0, A represents the probability of 0 being interpreted as a 1. Conversely, if the inverter input is at a 1, B represents the probability of 1 being interpreted as a 0.

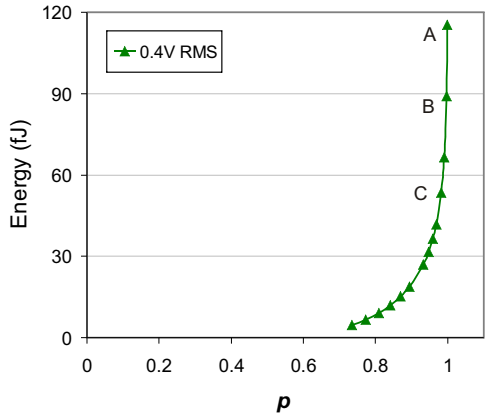


Figure 4: Energy-probability relationship of a PCMOS inverter designed in TSMC 0.25μm technology with a noise RMS of 400mV [27]. Energy consumption can be reduced by as much as 50% in exchange for roughly a 1% sacrifice in probability of correctness.

energy consumption and probability of correctness for PCMOS gates. Characterized extensively for an inverter [19, 20, 40], PCMOS operation allows slight concessions in accuracy to be traded for large savings in energy consumption. The Ep profile for an inverter operating a 400mV RMS, pictured for a TSMC 0.25μm process in Figure 4, shows that as much as 50% of energy consumption can be saved by sacrificing roughly 1% in probability of correctness.

Similarly, the relationship holds for more complicated gates. Shown in Figure 5 for a 24-transistor full adder implemented in TSMC 0.18μm technology, the Ep profile follows a quadratic relationship between energy and probability of correctness. Again, as much as

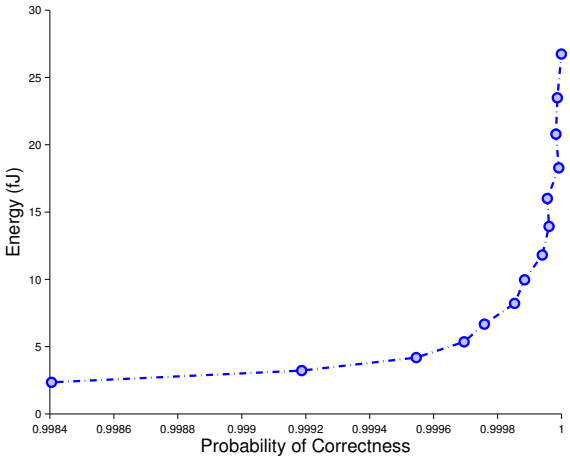


Figure 5: Energy-probability relationship of a PCMOS full adder designed in TSMC $0.18\mu m$ technology with a noise RMS of $150mV$. As was the case for the inverter in Figure 4, energy consumption can be reduced by as much as 50% in exchange for roughly a 1% sacrifice in probability of correctness.

50% of the energy consumption can be saved by sacrificing far less than 1% in probability of correctness.

2.5 Probabilistic Applications

From the application of PCMOS at the hardware layer it is apparent that probabilistic computing offers the potential to reduce energy consumption in exchange for accuracy. This exchange, however, requires that any application utilizing the underlying hardware be able to operate with the uncertainty introduced by PCMOS. Given this necessity, applications utilizing probabilistic algorithms are well suited to PCMOS operation due to their natural probabilistic operation.

Example 5: Consider a simple system intended to determine if there has been rain ($R = 1$). A Bayesian network might model this system with two input parameters: is the grass wet ($W = 1$) and is the sun out ($S = 1$). Empirical data for the likelihood of rain is then collected for each possible input state. From this empirical data it is determined that when the grass is wet and the sun is out there is a 2% chance that it has rained (the grass is more likely wet from a sprinkler system). Utilizing PCMOS, such a system could configure a NAND gate such that $R = \overline{W \cdot S}$ and set $p = 0.98$. In this configuration, 98% of the time the gate would evaluate to $R = 0$, inferring in most cases that there has not been rain if the grass is wet and the sun is out. □

Other examples of probabilistic applications include random neural networks that model the human brain, probabilistic cellular automata intended to model stochastic processes, and hyper-encryption that provides a provably secure encryption technique [16, 63]. These probabilistic applications rely on probability for correct operation. As such, they require a source of randomness to function and the quality of the random source will impact the resulting behavior of the application—low quality random sources can alter application behavior such as the correctness of Monte Carlo simulations or the strength of encryption schemes in hyper-encryption. Typically, this source of randomness is provided by pseudo-random number generators (PRNGs). A PRNG, however, is a complex solution to randomness that requires substantial silicon area for implementation and can lack in quality of randomness.

PCMOS, on the other hand, provides a source of randomness that is substantially better than PRNGs. Using statistical tests from a NIST suite, a PCMOS implementation passed 79% of the tests compared to 50% for a standard CMOS implementation [16]. Further, PCMOS provides this randomness for substantially less silicon area (and substantially less power). Systems considered yielded savings by orders of magnitude when implemented in PCMOS as compared to standard CMOS implementations: Bayesian networks, random neural networks, probabilistic cellular automata, and hyper-encryption [16]. Rather than avoiding noise as an impediment, these probabilistic applications capitalize on noise as a source of randomness.

2.6 Resilient Applications

Deterministic applications, unlike their probabilistic counter parts, are less obvious choices for probabilistic computing. By their very nature, deterministic applications are designed with the assumption of error-free operation. Despite this assumption, however, some of these deterministic applications exhibit resiliency against bit errors. While these applications are unable to functionally capitalize on the probability that PCMOS introduces, they are able to maintain operation in the presence of errors introduced through PCMOS. Because of this ability to tolerate probabilistic errors, these types of applications are referred to as *resilient applications.*

While the majority of all applications require determinism, only a subset of all deterministic applications exhibit resiliency. A banking application, for instance, has little margin for error. It is very unlikely that miscalculations in account balances or account routing would be well received by account holders. An image processing application, on the other hand, is much more tolerant to errors as long as they are well behaved. Miscalculations that result in slight deviations from the intended image are likely to pass unnoticed without adversely affecting a viewer's experience.

> **Example 6:** Considering an image processing application, images will typically be rendered using a color gamut of red, green, and blue (R,G,B) with the intensity of each color weighted from 0 to 255. In this color scheme white is defined as (R,G,B) = (255,255,255). Given an image intend to be viewed as pure white, a human viewer is unlikely to notice a white rendered as (R,G,B) = (255,250,255). In this case, resiliency in the image processing application is derived from color tolerances realized through human perception. □

Signal processing applications, in particular, are excellent candidates for resilient applications. Many signal processing applications are intended for human consumption. As such, a human viewer is the determiner of application quality. This allows the opportunity for human perception to filter noise that might be introduced through the application of PCMOS. As long as any errors result in deviations that come close to matching intended

results, a human observer is unlikely to notice the slight variations.

Adaptive algorithms are another class of applications that are well suited for resilient operation. Simulated annealing, for one, utilizes an iterative approach to global optimization by applying a random walk to a solution space. Global optimization is ensured by starting with large random jumps initially before reducing the jump distance slowly to achieve local optimization. Errors early in the algorithm, when jumps are large, are unlikely to have a significant impact on optimization results. As the algorithm reduces the iteration distance, error rates can be reduced allowing the local minimums to be calculated error free.

In adaptive filtering, the filter transfer function is adjusted automatically based on an error signal and an optimization algorithm. Occasional deviations in either the error or optimization calculations would cause the filter to momentarily diverge from the optimal filter coefficients. Assuming errors were infrequent, any errors would simply delay convergence to the optimal transfer function.

Vector quantization is another example of an adaptive algorithm well suited for resilient operation. Using vector quantization, data compression is achieved by mapping a data stream to a library of data symbols. Each symbol in the data library corresponds to a unique data sequence (a series of ones and zeros) and an optimal encoding minimizes the symbol sequence selected to represent the data stream. Applying PCMOS to the symbol selection algorithm might not select the optimal symbol sequence, however, accuracy is not impacted and the only penalty is a slight decrease in compression rates.

Finally, MPEG video compression subdivides video frames into macroblocks and searches for macroblocks within a frame sequence that are repeated across frames. As video tends to contain objects that are in motion, these macroblocks are likely to "move" from one frame to another. Compression is achieved by encoding a macroblock only once then predicting the location of that block in subsequent frames through a motion vector. Selection of a less than optimal macroblock/motion-vector pair will again cause a loss in compression, but at no cost to accuracy.

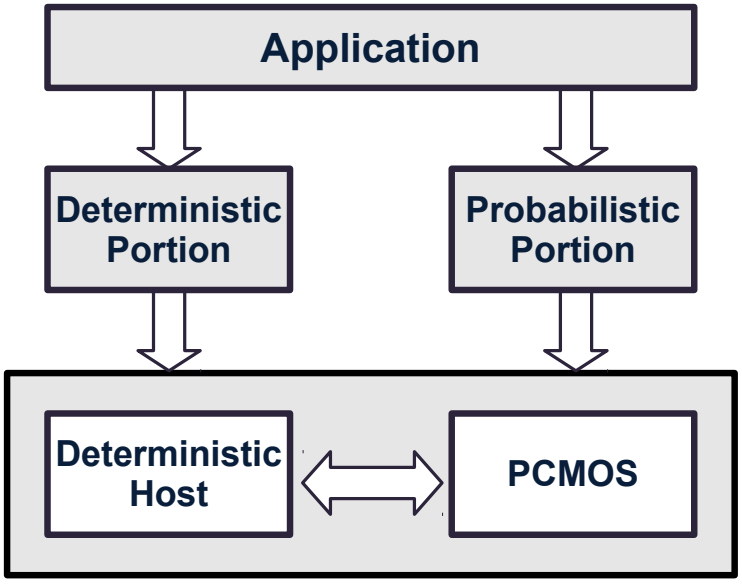


Figure 6: A Probabilistic System-on-Chip (PSoC) with a standard CMOS host processor and a PCMOS co-processor. Applications are partitioned into deterministic and probabilistic portions. Deterministic code is executed on the standard CMOS host processor and probabilistic code is executed on the PCMOS co-processor.

2.7 Probabilistic System-on-a-Chip

Regardless of the application, a Probabilistic System-on-a-Chip (PSoC) is envisioned as the architecture for PCMOS based systems. In both probabilistic and resilient applications, not all portions of the application can handle the probability introduced by PCMOS. Control logic, such as branches for instance, must execute deterministically to ensure proper program flow. To accommodate both deterministic and probabilistic program requirements, a PSoC provides a standard CMOS host processor and a PCMOS co-processor. Program code is then divided into deterministic and probabilistic portions. The deterministic portion of a program executes on the standard CMOS host processor and the probabilistic portion is offloaded to the PCMOS co-processor (Figure 6).

The PCMOS co-processor utilizes dynamic voltage scaling (DVS) to dynamically adjust voltage configurations, and associated error rates, to meet changing quality requirements. Applications, in turn, are provided with a lookup table of operational modes and their associated energy/probability profiles. Internally, the modes designate voltage configurations

that are applied to the PCMOS co-processor to achieve specific operational points. Externally, the operational modes provide applications with a tool set of predefined operating points that are optimized to trade accuracy for energy savings.

Example 7: A mobile telephony application, for instance, might have two calling modes: standard and low-battery. In standard mode voice quality is important and battery life is sacrificed for call quality. In low-battery mode battery life is important and call quality is sacrificed to extend battery life. Operating on a PSoC platform, the application might have access to the following PCMOS operational modes.

Mode	**Energy** ($pJ/clock$)	**Mean Squared Error**
0	43.4	0
1	35.7	$40,769$
2	30.9	$89,411$
3	25.6	$492,590$

Data represents configurations for a low-pass, finite-impulse-response filter (Section 4.7)

Given the available operating modes, standard calling mode might utilize the PCMOS co-processor in mode 1 to save some power while maintaining high voice quality. As battery life is depleted, however, eventually the application reconfigures the PCMOS co-processor for mode 3. This triggers the reconfiguration of the supply voltages for the PCMOS co-processor, saving additional power at the expense of a degradation in call quality. □

CHAPTER III

BIASED VOLTAGE OVERSCALING

3.1 Achieving Application Resiliency

Unlike their probabilistic counterparts, resilient applications are unable to utilize randomness at an algorithmic level. In direct contrast, these applications tolerate the randomness that is introduced through PCMOS to perform approximate, rather than absolute, calculations. This tolerance creates a trade off between application quality and energy savings. Better approximations result in higher levels of application quality, however, reductions in energy consumption lead to less accurate approximations. As a result, the energy savings available to a resilient application are limited by the amount of error that the application can tolerate for a given quality level.

Because energy savings are limited by application quality requirements, design for resilient applications is an exercise in minimizing energy consumption while maximizing the accuracy of approximations. With this goal in mind, a fundamental observation is that not all errors are created equal when it comes to approximations. The binary number system places weights on bit positions (bit i is weighted by 2^i) creating a hierarchy of bit significances. Errors at bits with a higher significance lead to more significant errors. Shown in Figure 7 for a 16-bit adder, an error on the sum bit of a full adder at bit 0 results in an error magnitude of 1. A bit error on the same sum bit of a full adder at bit 15, however, results in an error magnitude of 32,768.

At the application level, error magnitude translates to approximation accuracy and ultimately application quality. Maintaining application quality necessitates that any bit errors result in small deviations from the intended results. As such, the bit position where errors are generated is important.

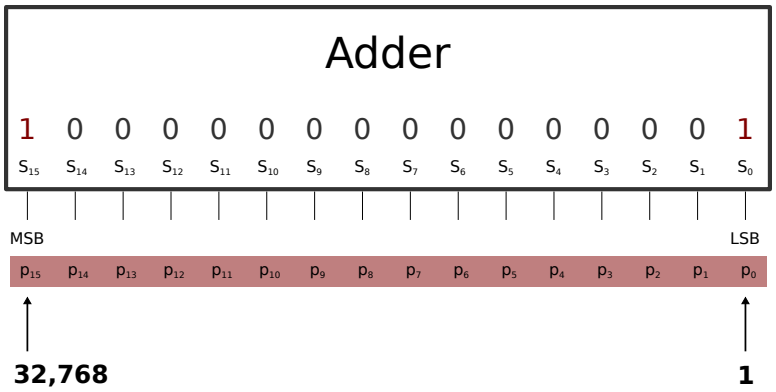


Figure 7: The bit position where an error occurs is significant. Shown here for a 16-bit adder, an error occurring at the sum of the full adder at bit 0 results in an error magnitude of 1. A similar error occurring on the sum bit of the full adder at bit 15 results in an error magnitude of 32,768.

> **Example 8:** Revisiting image processing from Example 6, assume a pixel is intended to be rendered as pure black, (R,G,B) = (0,0,0). As before, the color intensity is at an 8-bit resolution with weights ranging from 0 to 255. If an error were to be generated at bit zero when calculating the color intensity for red, this would result in a bit flip ($0 \rightarrow 1$) causing the rendered pixel to be (R,G,B) = (1,0,0), a color very close to pure black. Conversely, if an error were to be generated at bit six the resulting pixel would be rendered as (R,G,B) = (128,0,0), something much closer to red. □

While tolerances will vary from application to application, from Example 8 it is clear that high-order errors can be catastrophic. Any errors that occur must be constrained to low-order bit positions to minimize the impact on approximation accuracy. Unfortunately, applying conventional voltage overscaling uniformly reduces supply voltages across all bit positions. This, in turn, sacrifices accuracy uniformly across all bit positions meaning that errors are equally as likely at high-order-bit positions as they are at low-order-bit positions. As a result, application quality degrades rapidly as supply voltage is scaled past the point where errors begin occurring. Shown in Figure 8, an example of uniform voltage scaling is used for H.264 video decoding. Errors occurring at high order bit positions cause the associated pixels to saturate to blacks and whites, resulting in an extremely pixelated image.

Figure 8: Uniform voltage scaling applied to H.264 video decoding. While the voltage overscaling substantially reduces power consumption, high-order-bit errors quickly degrade application performance.

Biased Voltage OverScaling (BIVOS) addresses this problem by capitalizing on bit significance. With BIVOS, voltage overscaling is biased such that energy consumption is distributed by the significance of bit positions. Elements that calculate the most significant bit positions receive nominal supply voltages to ensure no errors are generated. Voltage levels are then gradually decreased with the elements that calculate low-order bit positions receiving the lowest voltage levels. This results in a probability distribution where errors are more likely to be generated at bit positions that contribute the least to approximation accuracy.

BIVOS then dictates that $V_i \geq V_{i-1}$. Given this requirement, a distinct voltage level could potentially be assigned to the elements at each bit position. Shown in Figure 10 (a), an n-bit adder receives n distinct voltage levels with all elements that compute S_i receiving V_i. While such a configuration could lead to an optimal solution, practical considerations place limitations on the number of distinct supply voltages available to a design. In cases where the number of available voltage levels are limited, binning can be employed to reduced the number of voltage sources required. With binning, individual bits are grouped with neighbors to form b bins. For the purposes of biasing, all bit positions within a bin are considered to have equal significance and receive the same supply voltage. Bins need not be equivalently sized and the only requirement is that bins be formed from adjacent elements.

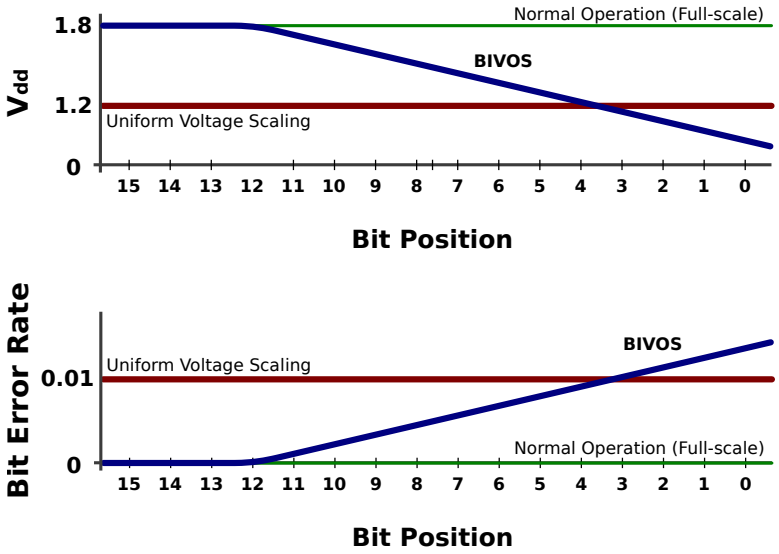


Figure 9: Where typical voltage scaling lowers the voltage uniformly accros all bit positions, biased voltage scaling distributes voltages unevenly by bit significance. In turn, the resulting errors due to voltage overscaling are biased to low order bit positions where the impact of any bit errors is reduced.

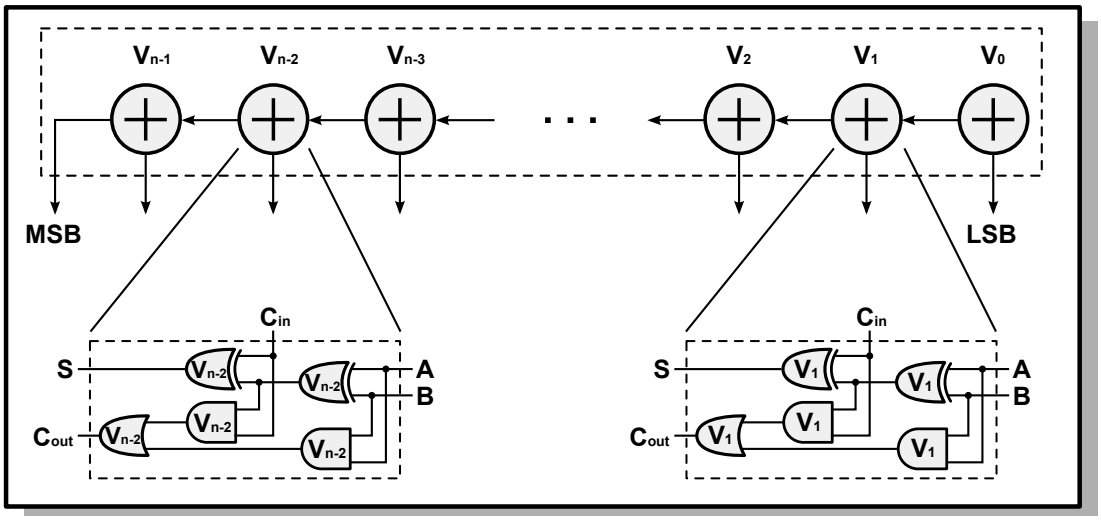
The resulting bins are then biased such that $V_b > V_{b-1}$. Shown in Figure 10 (b), the same n-bit adder is biased using b bins.

Regardless of the distribution employed, it is possible for errors to occur at high-order bit positions. Shown in Figure 11, a sensitized carry chain can result in a low-order error propagating to a high-order output. While this is technically a high-order bit error, the propagation has also resulted in offsetting errors along the carry path. The result is that error magnitude is constrained by the significance of the bit position where the error was generated.

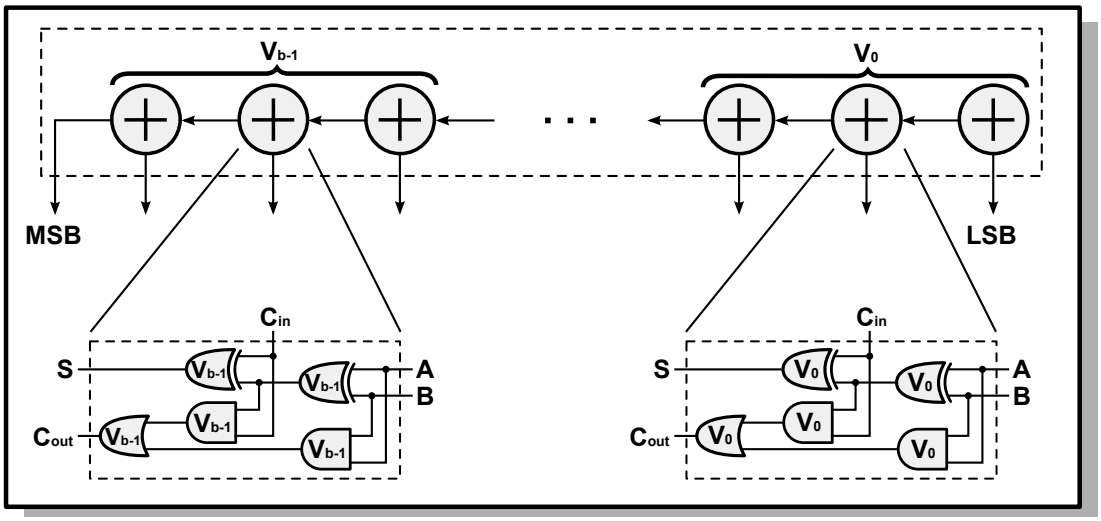
In turn, BIVOS limits error magnitudes by biasing errors to low-order bit positions. Energy consumption is biased to elements that contribute most to approximation accuracy and errors are biased to elements that contribute least to error magnitude. Ultimately this leads to minimizing energy consumption while maximizing approximation accuracy.

3.2 Challenges of Biased Voltage Overscaling

An obvious prerequisite to biasing via multiple voltage levels is the availability of multiple voltage sources. Binning relaxes the need for a high number of sources, however, BIVOS



(a)



(b)

Figure 10: Biased voltage scaling (or BIVOS) example for (a) an n-bit, ripple-carry adder employing BIVOS with distinct voltage supplies for each 1-bit full adder and (b) an n-bit, ripple-carry adder employing BIVOS with voltage binning. In each case, the voltage at a bit position is applied to all elements (INV, AND, OR, etc.) at that bit position. Where a fully biased BIVOS adder might require n distinct voltage levels, binning reduces the number of voltage sources required such that $b < n$.

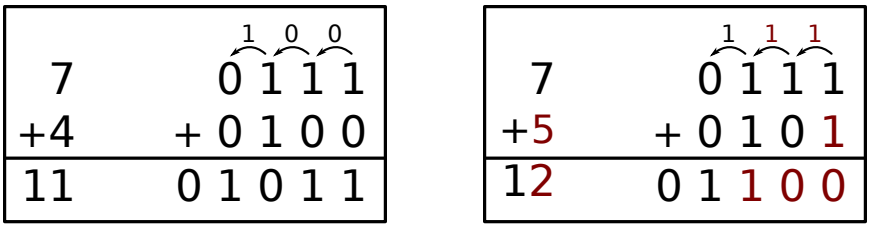


Figure 11: Carry propagation along a sensitized carry path: (a) correct result of the addition function ($7+4=11$) and (b) propagation of an error incurred at bit 0 resulting in incorrect bits at bit 1 and bit 2 in addition to the original error. Despite propagation down an activated carry chain, the resulting error magnitude is equivalent to the significance of bit position where it occurred ($12-11=1=2^0$).

fundamentally requires more than one voltage level. Not an uncommon problem, a typical method for providing multiple voltage levels is through DC-DC step-down converters. In each step-down converter there is a power penalty due to less than 100% efficiency in the voltage conversion. In the case of conventional design, solutions are generally limited to two or three distinct voltage levels. The BIVOS approach, on the other hand, has no such constraints. Accordingly, increasing the number of step-down converters for a BIVOS solution would initially appear to compound the power penalty by adding additional converters with additional inefficiency. This is not the case, however, as adding step-down converters to a solution reduces the loading on pre-existing step-down converters. Global loading is spread over multiple step-down converters that are now sharing the inefficiency of the original step-down converters.

The primary drawback to providing multiple voltage sources is area cost. Each additional step-down converter, whether placed on die or off, increases the overall area consumption of the power solution. This becomes an issue in designs with size limitations, which is the case in most embedded designs where power consumption is a primary consideration. In turn, BIVOS requires design time trade-offs in terms of area and power efficiency.

Further complicating the area cost of multiple voltage sources, is the associated routing necessary for power delivery. Each additional voltage level requires a separate on-die, power-distribution grid. In [41] it was shown, when compared to a single V_{dd} design, a dual V_{dd} design could reduce the decoupling capacitance budget with no area overhead (and in some

cases reducing wire congestion). This is primarily attributed to the reduction in current draw from each individual power supply achieved by the dual V_{dd} design. By reducing current draw the demands on the power distribution grid are reduced, allowing for lighter weight supplies and a reduction in the sizing of power rails. This solution, however, considers only a dual V_{dd} power distribution. There is a limit to the extent that power rail reduction is possible regardless of the current draw of an individual grid. After power rail scaling reaches technology minimums, any additional voltage grids will add to area overhead.

Once multiple voltage planes have been established, communication between the planes becomes an issue. Attempting to drive high-voltage gates with signals from low-voltage gates results in static current flow as transistors never fully switch off. A standard solution to this problem is to insert level converters at voltage boundaries. Level converters, however, dissipate additional power and without judicious use can overwhelm any power savings gained through BIVOS. Further, they can complicate layout due to the need for both high and low voltage access to convert signals. In [68], level converters requiring only a single, high-voltage supply are used to simplify placement. This is taken a step further by using level-converter-less, dynamic circuits for FPGA design in [21]. As an alternative, inverters can be inserted at voltage transitions to limit static current to a single transistor pair. While this approach does not yield zero static current, it does simplify design and reduce static current draw allowing BIVOS configurations to yield reductions in energy consumption.

3.3 Reduced Precision as an Alternative

Given the extra cost associated with implementing a BIVOS solution, an obvious alternative is a reduction in precision. Rather than employing voltage overscaling with multiple voltage sources, a reduced-precision solution simply removes bits altogether. This can be accomplished with a reduced-hardware solution or with a scheme that powers down individual bits. A reduced-hardware solution reduces circuit area, however, it limits flexibility at run time. A power down scheme alternatively saves no area, but does allow for dynamic tuning to meet application quality requirements. Regardless of the implementation, a reduced-precision solution introduces error not unlike a PCMOS solution.

In a reduced-precision solution, error is introduced by an inability to accurately represent real numbers within the digital system. Conventional digital computation represents real numbers in one of two binary formats: fixed-point or floating-point. Of the two, floating-point representation offers greater precision and a dynamic number range. This increased precision and range, however, comes at the expense of increased hardware design complexity, silicon area, and power consumption. As the work presented here focuses on low-power computing, and fixed-point representation is typically employed in low-power applications, floating-point representation is not considered.

In fixed-point representations, a binary number is divided into integer and fractional parts by a binary point [59]. The decimal value of an n-bit, fixed-point number, D, consisting of B integer bits and b fractional bits is equal to the sum of the bit values at each bit position multiplied by the significance of the given bit position (Equation 8).

$$D = \sum_{i=-b}^{B-1} a_i * 2^i \tag{8}$$

The binary point is assumed to be *fixed* at a specific bit position, hence *fixed-point* representation. As the binary point is purely a programming construct, it is not visible to the underlying hardware. This allows hardware designed to implement integer addition and multiplication functions to operate on fixed-point numbers [54].

> **Example 9:** Considering the number $1010_{\triangle}111$, where $_{\triangle}$ represents the binary point, the portion of the number to the left of the binary point represents the integer portion, 1010, while the portion of the number to the right of the binary point represents the fractional portion, 111. □

The range of non-negative, integer numbers η that can be represented by a n-bit, fixed-point number is limited to

$$0 \leq \eta \leq 2^n - 1. \tag{9}$$

The range of non-negative, fractional numbers η that can be represented by a B-bit, fixed-point number is limited to

$$0 \leq \eta \leq 1 - 2^{-n}. \tag{10}$$

In both cases, the range of representable numbers is fixed to a minimum η_{min} and a maximum η_{max} value. The dynamic range is then given by $R = \eta_{max} - \eta_{min}$ and the resolution of the representation is given by

$$\Delta = \frac{R}{2^n - 1}, \tag{11}$$

where Δ represents the quantization level [54]. Any real number that cannot be represented by the available resolution of the fixed-point representation in use is therefor rounded, or quantized, and the resulting error introduced by the quantization process $\mathcal{Q}(X)$ affects the accuracy of subsequent calculations.

> **Example 10:** Assuming an unsigned two-bit, fixed-point representation with two fractional bits and zero integer bits, the representation would yield a range $0 \leq \eta \leq 0.75$ with a quantization level $\Delta = 0.25$. As such, attempting to represent a real-value number of 0.40 in this fixed-point representation would require the number be rounded up to an approximation of 0.50 or down to an approximation of 0.25. Employing a nearest neighbor rounding scheme, the real-value number would be rounded to 0.50 with a quantization error of 0.10. □

Error introduced through quantization is limited by the quantization step size for a particular fixed-point implementation and dictated by the rounding scheme employed. In the case of a nearest neighbor rounding scheme, error magnitude $\mathbf{MAG}_E$ is bound by

$$\frac{-\Delta}{2} < \mathbf{MAG}_E \leq \frac{\Delta}{2}. \tag{12}$$

Similarly, a truncation rounding scheme (round down) bounds $\mathbf{MAG}_E$ to

$$-\Delta < \mathbf{MAG}_E \leq 0. \tag{13}$$

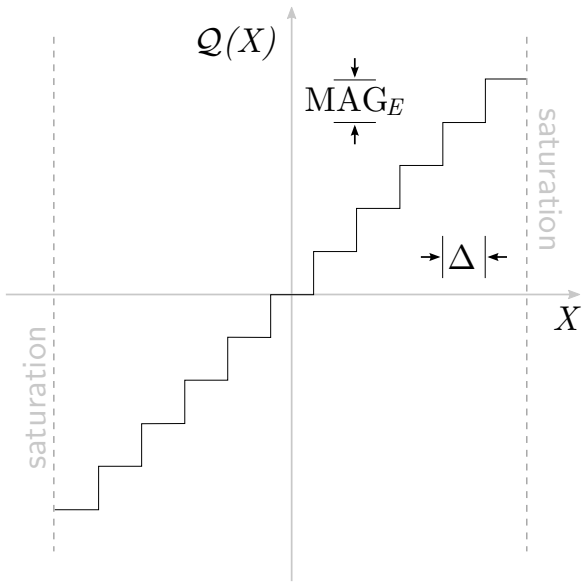


Figure 12: Output versus input for a linearly spaced, nearest neighbor quantizer. Quantization error is limited to $\pm\Delta/2$ by the quantization step Δ.

While quantization is a non-linear process (Figure 12), the error introduced through quantization is typically modeled as the addition of noise (a linear process). Making the following assumptions allows for this simplified model [59]:

- $\mathbf{MAG}_E$ is uniformly distributed over the error range
- $\mathbf{MAG}_E$ is white noise
- $\mathbf{MAG}_E$ is uncorrelated with X
- $\mathbf{MAG}_E$ is a zero-mean, stationary process

Expected noise power can then be calculated by integrating over the error probability density function (Equation 14) [59].

$$\mathbf{MSE}(\mathcal{Q}(X)) = \int_{-\infty}^{\infty} X^2 p_e(X)\,\mathrm{d}X \tag{14}$$

Based on the assumption that quantization error is uniformly distributed, the probability density function yields $p_e(X)$ equal to $1/\Delta$ over the error range (Figure 13).

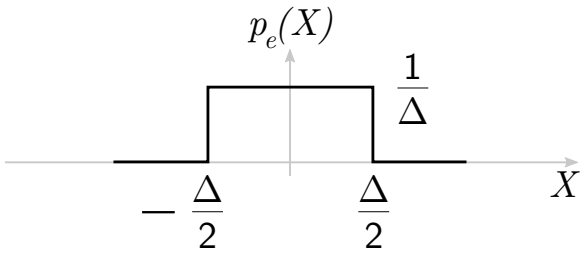


Figure 13: Probability density function a nearest neighbor quantizer. Error is limited to $\pm\Delta/2$ with a uniform probability distribution equal to $1/p_e(X)$.

Substituting $p_e(X)$ equals $1/\Delta$ for the range $-\Delta/2$ to $\Delta/2$ (a nearest neigbhor rounding scheme) in Equation 14 yields

$$\mathbf{MSE}(\mathcal{Q}(X)) = \int_{-\Delta/2}^{\Delta/2} X^2 \frac{1}{\Delta}\,\mathrm{d}X \tag{15}$$

$$= \frac{\Delta^2}{12}.$$

Equation 15 is then dependent only on the quantization step size Δ. By assuming uniform, white, uncorrelated, and zero-mean noise (which is generally valid); quantization error can be estimated purely as a function of quantization step size, independent of X.

Example 11: As in Example 10, assume a two-bit, fixed-point representation with two fractional bits ($b = 2$) and zero integer bits ($B = 0$) employing truncation rounding. Equation 14 then reduces to

$$\mathbf{MSE}(\mathcal{Q}(X)) = \int_{-\Delta}^{0} X^2 \frac{1}{\Delta}\,\mathrm{d}X$$

$$= \frac{\Delta^2}{3}$$

and substituting Δ equals 0.25 yields

$$\mathbf{MSE}(\mathcal{Q}(X)) = \frac{0.25^2}{3} = 2.1\text{x}10^{-2}.$$

□

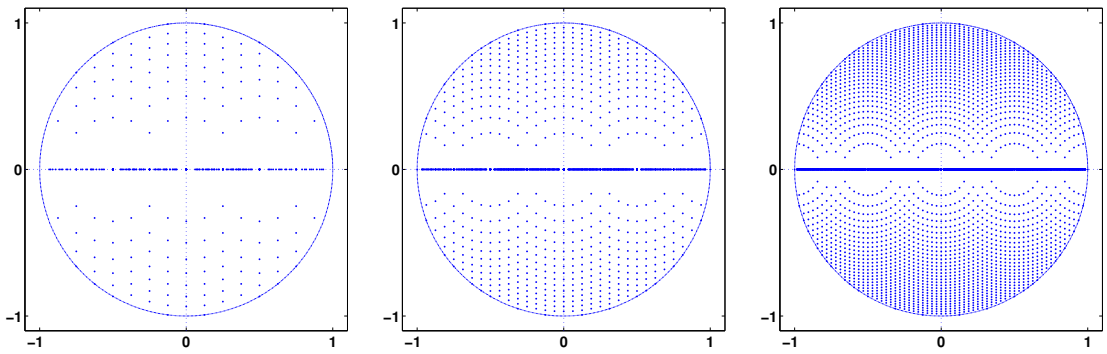


Figure 14: All stable pole locations for (*a*) 4, (*b*) 5, and (*c*) 6-bit, fixed-point second-order polynomials. Removing a single bit from a fixed-point solution drastically reduces the number of stable pole locations.

In the context of a reduced-precision power saving technique, quantization matches an m-bit input to an n-bit, fixed-point circuit (where $m > n$) by reducing the bit resolution. Quantization error is then reflected in the circuit inputs as additive noise and the circuit transfer function H determines the resulting output noise for a reduced-precision operation $\mathcal{O}_{\mathcal{R}}$ as

$$\mathbf{MSE}(\mathcal{O}_{\mathcal{R}}) = \mathbf{MSE}(\mathcal{Q}(X))H^2. \tag{16}$$

Applied to filtering, frequency response can be extremely sensitive to coefficient quantization noise. Because a reduced-precision solution reduces the number of discrete values a number system can represent, locating stable poles within a reduced operating space can be challenging. Shown in Figure 14, the number of stable poles for a particular solution can vary drastically based on bit-width employed. As small changes to coefficient values can cause poles to move outside the stable region, coefficient quantization noise can result in an unstable filter [59].

CHAPTER IV

COMPARING BIASED VOLTAGE OVERSCALING AND REDUCED PRECISION

4.1 Standard Cell Library

To compare the area requirements for a BIVOS-based circuit design and a more standard CMOS technique, such as reduced precision, three commonly used adder architectures were considered: ripple carry, carry select, and block propagate. The first step in building these architectures was to complete a small library of standard cells. As a starting point, a 24-transistor full adder was designed (Figure 15) as discussed in the Weste and Harris text CMOS VLSI Design [86]. Layout was performed in Cadence Virtuoso using the NCSU Cadence Design Kit for TSMC $0.18\mu m$ Regular ($0.20\mu m$) technology. Transistors were sized and placed in accordance with the datapath example from the Weste and Harris text. Of note in this particular design, wiring is completed in polysilicon and metal one, with metal two reserved for horizontal routing. In order to compress the design footprint, Weste and Harris have placed the supply, V_{dd}, and ground, Gnd, rails in metal two allowing placement above the cell. The resulting full adder implementation is $12.8\mu m$ by $4.8\mu m$ with an area of $61.44\mu m^2$.

In addition to a full adder (FA), the three adder architectures require inverter (INV), multiplexer (MUX), exclusive-or (XOR), not-and (NAND), partial and-or-invert (AOI), and level converter (LC) standard cells. With the 24-transistor, full-adder design serving as a reference for cell pitch, the remaining cells were sized to match. Conventional transistor sizing was adopted for the INV with a 2/1 width ratio of pMOS to nMOS. Matching inverter pitch to the 24-transistor full adder results in dimensions of $3.65\mu m$ by $4.8\mu m$ with an area of $17.52\mu m^2$. The two-input MUX and XOR circuits implement identical logic and are duplicated only to ease signal identification. In order to match inverter-input capacitance, these cells are also sized with a 2/1 pMOS-to-nMOS ratio with minimum

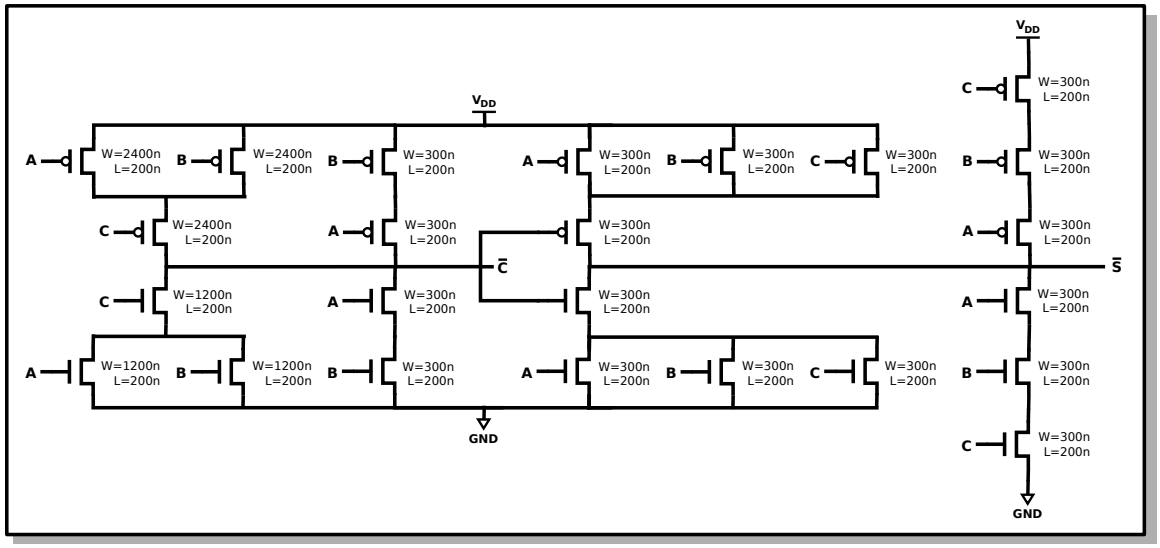


Figure 15: Transistor schematic for a 24-transistor, full adder with transistor sizing to minimize propagation delay through the carry output [86].

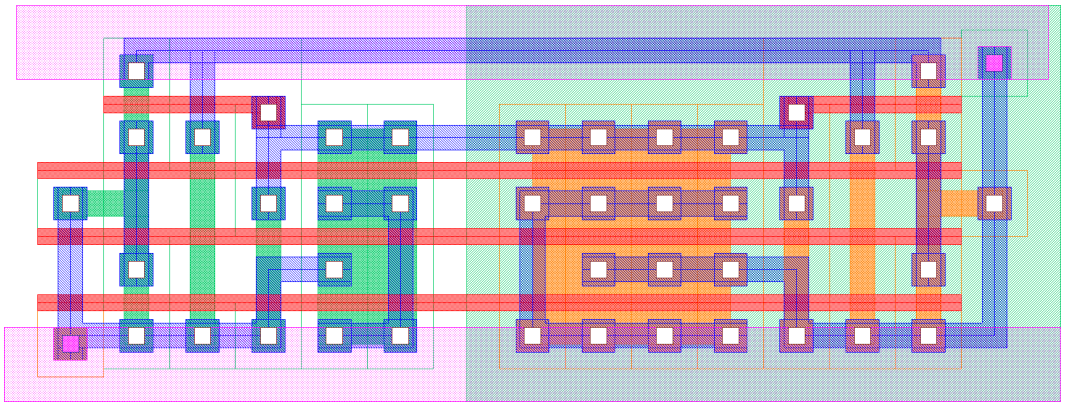


Figure 16: VLSI layout for the same 24-transistor, full-adder design shown in Figure 15. In order to compress the design footprint, the supply, V_{dd}, and ground, Gnd, rails have been placed in metal two allowing placement above the cell [86]. Implemented in TSMC $0.18\mu m$ Regular technology, the resulting cell is $12.8\mu m$ by $4.8\mu m$ with an area of $61.44\mu m^2$.

Table 1: Standard Cell Area Consumption

Cell	**Height** (μm)	**Width** (μm)	**Area** (μm^2)
INV	4.8	3.65	17.52
MUX	4.8	3.95	18.96
XOR	4.8	3.95	18.96
NAND	4.8	3.80	18.24
NAND4	4.8	5.10	24.48
AOI	4.8	4.65	22.32
LC	4.8	3.60	17.28
FA	4.8	12.8	61.44

sized nMOS transistors. The 4-input NAND gate extends a standard 2-input NAND to 4 inputs. Again, to match inverter-input capacitance, pMOS transistors are minimum size with nMOS transistors sized at a 2/1 ratio to pMOS transistors. The AOI cell implements the equation $\overline{A + (B \cdot C)}$. Transistors are again sized at a 2/1 pMOS-to-nMOS ratio with nMOS transistors minimum sized. Unlike the other cells, the level converter performs no logical function and simply translates a low voltage signal to a high voltage signal. In this case, transistors are sized through experimentation to minimize power consumption with a 3/8 pMOS-to-nMOS ratio and minimum sized pMOS transistors.

The resulting cell implementations are shown in Figures 80 through 86 of Appendix A and are detailed in Tables 1 and 2. All cells are roughly the same size with equivalent worst case logical effort of two (excluding the base line inverter, level converter, and full adder). Of these, only the AOI cell is unbalanced with a single input exhibiting a logical effort of one on the pull-down network.

4.2 Conventional Circuit Layout

As a baseline, each of the three adders were first designed for standard CMOS operation with the goal of minimizing area. Working from the standard-cell library, the full-adder outputs, sum ($\overline{S}$) and carry out ($\overline{C_{out}}$), were left in their inverted state per the design in the Weste Harris text [86]. By doing so, sequential full adders in an adder datapath are able to alternate between positive and negative logic. Inverters are only inserted where

Table 2: Standard Cell Design

Cell	**Input Capacitance** (C_{inv})	**Resistance** (R_{inv})	**Logical Effort**
INV	1	1	1
MUX	1	2	2
XOR	1	2	2
NAND	1	2	2
NAND4	1	2	2
AOI	1	2	2

necessary to match input/output bits to the appropriate logic (Figure 17). The result is a reduction in transistor count and an associated improvement in performance. This type of alternating logic is adhered to whenever possible, allowing the inversion of internal signals and accounting for input/output inversion where necessary.

The ripple-carry adder is the simplest design of the three considered and is implemented as a set of full adders in series (Figure 17). As is the case for all three architectures, the datapath for the ripple-carry adder is eight bits wide (since the designs are based on standard cells, datapaths larger than eight bits simply require tiling of 4-bit blocks). Previously noted, full adders alternate between positive and negative logic and input/output inversion is accounted for with added inverters. The resulting ripple-carry adder, shown in Figure 87 of Appendix B, is $36.6\mu m$ wide with a height of $20.1\mu m$ and an area of $735.66\mu m^2$.

More complicated than the ripple-carry adder, the block-propagate adder is comprised of two 4-bit, ripple-carry adder blocks. Propagate logic is calculated external to the full adders within each ripple-carry adder and a single multiplexer determines propagate or carry for each block. Elements are divided across four, two-bit rows to create a roughly square design. The resulting block-propagate adder is $63.5\mu m$ wide with a height of $24.4\mu m$ and an area of $1549.40\mu m^2$ (Figure 88, Appendix B).

Again, more complicated than the ripple-carry adder, the carry-select adder contains multiple 4-bit, ripple-carry-adder blocks. The first block is comprised of a standard 4-bit, ripple-carry adder. The second set contains a pair of 4-bit, ripple-carry adders. The carry-in

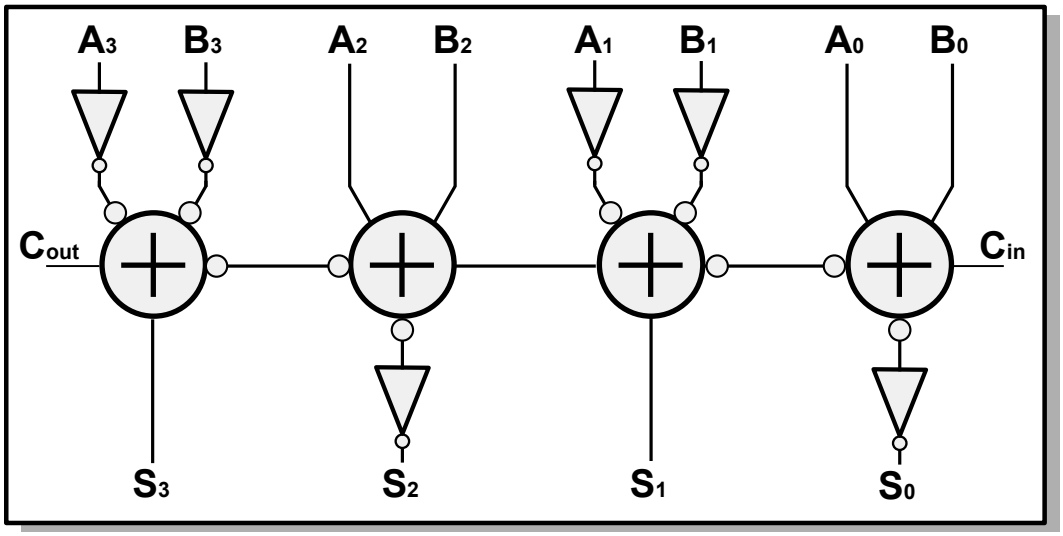


Figure 17: Ripple-carry adder implementation with alternating positive and negative logic. At bit 0 the full adder operates on positive logic (A_0, B_0, and C_{in}) generating negated outputs ($\overline{S_0}$ and $\overline{C_0}$). The sum output is inverted to generate S_0, however, the carry output is left in an inverted state. At bit 1 the full adder operates on negative logic ($\overline{A_0}$, $\overline{B_0}$, and $\overline{C_{in}}$) generating positive outputs (S_0 and C_0) without the need of inverters. When compared to a more conventional design, every two-bit pair in an alternating logic implementation removes both inverters on the carry outputs of each full adder and allows one-inverter to be saved on the sum output of the negative-logic full adder. This is at the expense of two additional inverters on the A and B inputs of the negative-logic full adder. The result is a net gain of one less inverter (a standard design would require four inverters for the two sum and two carry outputs) and a reduction of two inverters along the critical path of the carry chain.

bit of one ripple-carry adder is hard wired to zero and the carry-in bit of the other ripple-carry adder is hard wired to one. Multiplexers select between the two sets of sum bits based on the carry-in bit to the second block (or conversely the carry-out bit of the first block). Shown in Figure 89 (Appendix B), the elements are arranged in six, two-bit rows with the bottom four containing the carry-select logic in addition to the requisite full adders. Remaining components are placed in line to minimize area. The resulting carry-select adder is $43.9\mu m$ wide by $31.4\mu m$ tall, with an area of $1378.46\mu m^2$.

4.3 BIVOS Circuit Layout

For BIVOS design, the starting point was again the original standard cell library. As previously noted, a BIVOS design requires multiple supply-voltage levels. To accommodate multiple voltage levels the V_{dd} rail lengths were reduced by $0.15\mu m$ at both horizontal cell edges on each standard cell. This creates a minimum metal spacing of $0.3\mu m$ when adjacent cells are abutted to form a datapath, allowing for independent V_{dd} rails in each cell (Gnd rails are left unaltered as a common ground is required for operation). Should any adjacent cells operate at the same voltage, as in the case of standard CMOS design, broken V_{dd} rails are rejoined during datapath layout (Figure 18).

Utilizing the standard cell library, full adder outputs are in an negative state. The standard datapaths utilize this structure to remove inverters from the critical path. Employing the BIVOS technique, however, requires further consideration. Previously discussed, BIVOS relies on biasing bit errors to low-order bit positions using multiple supply voltages. Mentioned in several works, this creates a voltage mismatch at supply voltage boundaries [11, 12, 13, 85]. This results in static current flow when a low voltage device attempts to drive a high voltage device. The solution typically employed in these works is to insert level converters between voltage boundaries. Level converters consume additional power, however, and can quickly overwhelm any savings realized through BIVOS. As an alternative, inverters can be used to mitigate static current flow to a single transistor pair while avoiding the additional area, and power, overhead incurred with the use of level converters. This solution does have the drawback of limiting the biasing of adjacent voltage planes

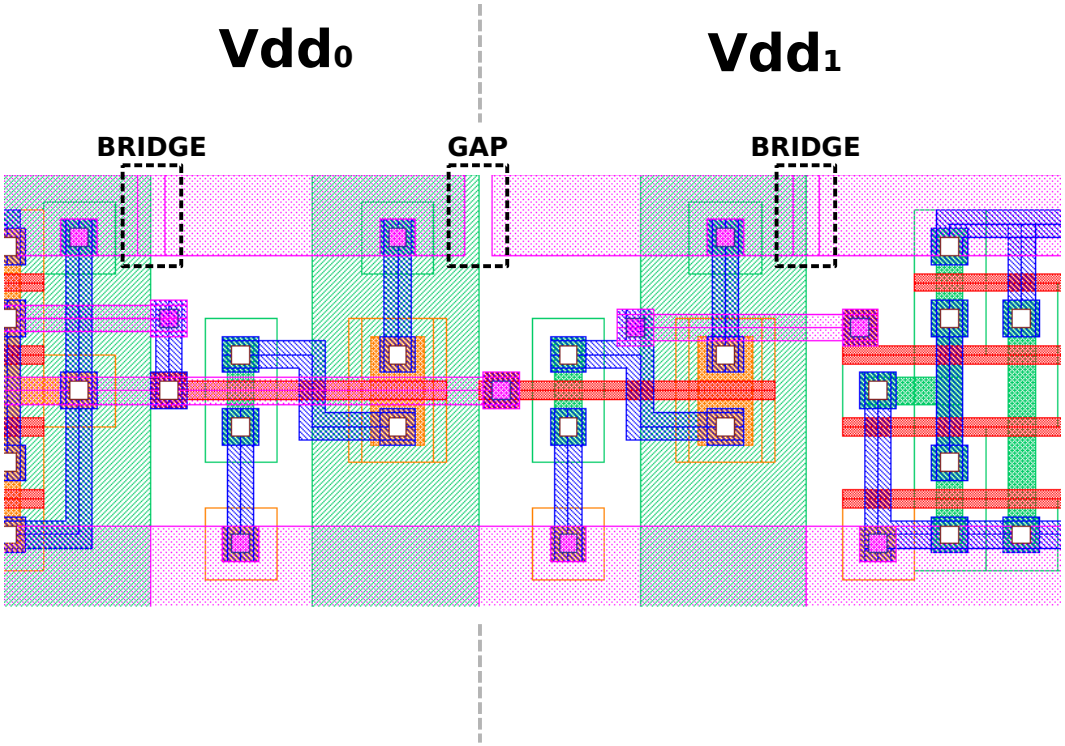


Figure 18: Voltage boundary at two bits for an adder architecture. V_{dd} rails for standard cells operating at the same voltage level are joined with the highlighted bridges. V_{dd} rails for standard cells operating at different voltage levels are left disconnected via the highlighted gap. Shown here for bits 0 and 1 of a ripple-carry adder, bit 0 is biased to V_0 and bit 1 is biased to V_1 with the carry out inverter from bit 0 operating at V_1 to mitigate static current.

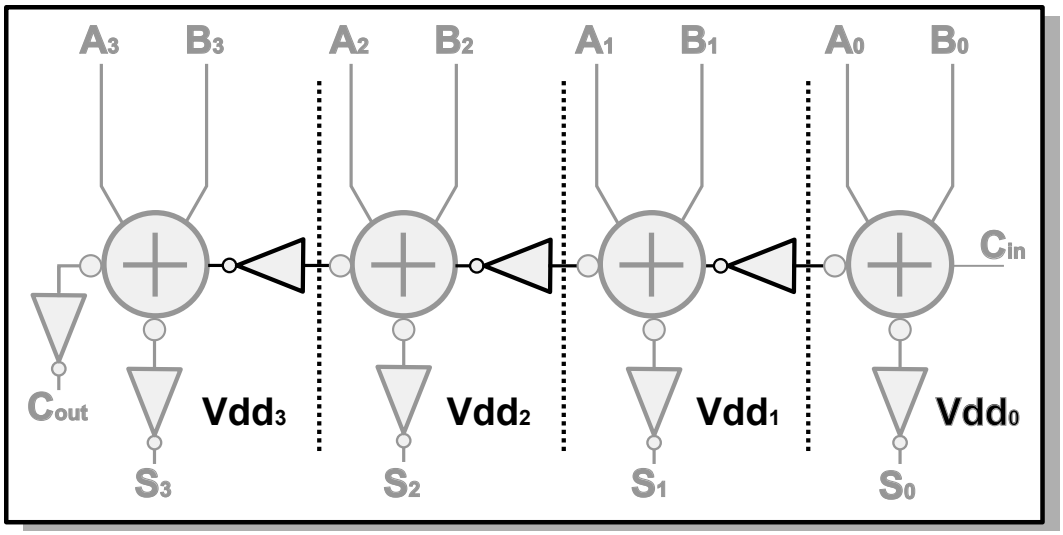


Figure 19: PCMOS based ripple-carry adder implementation with inverters acting as level converters at voltage boundaries. The addition of inverters along the carry chain makes the use of alternating logic impractical.

to less than the threshold voltage (anything greater than the threshold voltage can cause enough static current flow to overwhelm savings), however, it eliminates the need for formal level conversion.

BIVOS designs employing inverter-level conversion capitalize on the negative state of full-adder outputs to require only a single inverter, inserted between full-adder pairs, for "level conversion" at voltage boundaries (Figure 19). As the voltage differential between adjacent voltage planes is limited by threshold voltage, these designs require several distinct voltage levels (and added inverters for level conversion) to achieve a significant voltage reduction. The additional inverters negate any transistor savings made possible by alternating logic, making a positive logic design more practical. Designs employing traditional level conversion (Figure 20), alternatively, can support voltage differentials up to one half of the high voltage level at voltage boundaries. As a result, far fewer distinct voltage levels are required to realize a substantial voltage reduction. Limiting biasing to two distinct voltage levels then reduces the the overhead associated with the addition of level converters. This allows these designs utilize the same alternating logic employed for the standard datapaths with the addition of a single inverter and level converter at voltage boundaries.

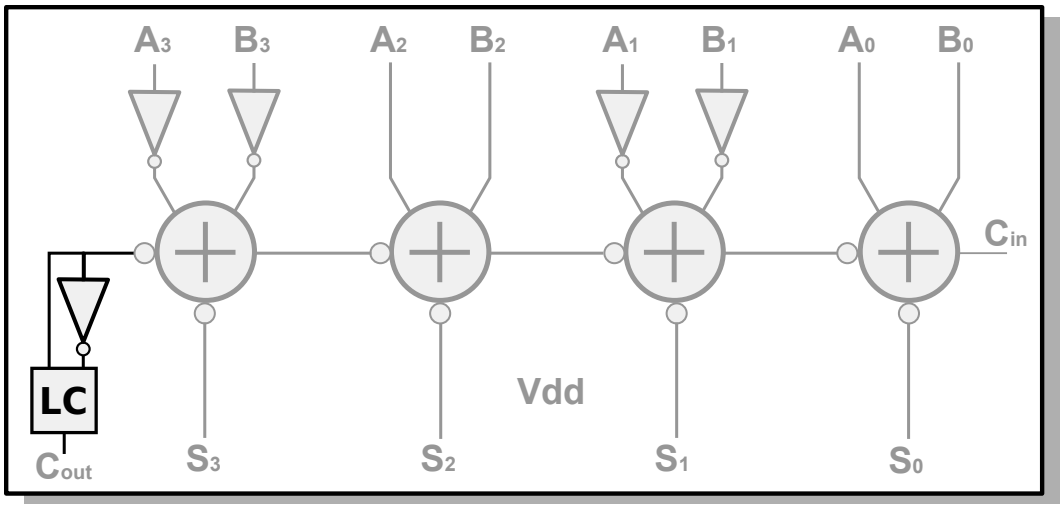


Figure 20: PCMOS based ripple-carry adder implementation with traditional level converters at voltage boundaries. By applying voltage binning in conjunction with traditional level converters, a reduction in transistor count through the application of alternating logic is possible.

Beyond requiring voltage conversion at voltage boundaries, BIVOS design adds an additional requirement that multiple voltage levels be routed to appropriate biasing positions. Designs minimize the resulting area impact by capitalizing on the fact that BIVOS distributes supply voltage by bit position. As such, aligning bit positions vertically across datapaths allows for voltage planes to be routed vertically in metal three (as many as one supply voltage for each bit position) and a single, vertical V_{dd} line can supply each cell along a vertical bit position. The result is that routing is achieved with minimal routing overhead within the datapaths and any area penalty incurred for the addition of voltage planes is limited to the routing from the voltage supply to the datapaths.

Following these guidelines, the three original adder architectures were modified to allow BIVOS operation utilizing both inverter level conversion (Figures 90, 91, and 92 in Appendix C) and traditional level converters (Figures 93, 94, and 95 in Appendix D). Designs employing inverter level conversion allowed for an independent supply voltage at each bit position. Designs employing traditional level converters where divided into two, four-bit voltage bins for biasing. Where adjacent cells operate on the same bit position, V_{dd} rails were joined to establish a common bias for the bit. As individual supply voltages were routed vertically in metal three, they added zero area overhead to the datapath design. The only

Table 3: Area Impact of PCMOS Versus Standard Circuit Design

Implementation	**Architecture**	**Width**(μm)	**Height**(μm)	**Area**(μm^2)	**Penalty**
Standard	Ripple-Carry	36.6	20.1	735.66	—
	Block-Propagate	63.5	24.4	1549.40	—
	Carry-Select	43.9	31.4	1378.46	—
PCMOS INV	Ripple-Carry	40.2	24.6	988.92	34%
	Block-Propagate	75.4	25.3	1907.62	23%
	Carry-Select	46.9	34.8	1632.12	18%
PCMOS LC	Ripple-Carry	40.2	21.6	868.32	18%
	Block-Propagate	67.3	24.6	1655.58	7%
	Carry-Select	43.9	32.6	1431.14	4%

area penalty was due to the necessary inclusion of inverters, acting as "level conversion," and level converters between voltage islands.

Shown in Table 3, the area penalty for employing the BIVOS technique can be non-trivial. Designs employing inverter level conversion suffer the worst area penalties. While these designs only require an additional inverter for each voltage conversion, the number of voltage levels required coupled with the added inverters necessitate the use of positive logic. The result is a significant area overhead for designs utilizing inverter level conversion. Designs employing traditional level converters, on the other hand, suffer a minimal area penalty for most architectures. Limiting voltages to two distinct levels allows the use of the same alternating logic utilized in the standard designs and only requires the addition of a single inverter/level-converter pair. As a result, the area penalty for employing a BIVOS design with traditional level conversion is less than 10% for two of the three architectures.

In the case of the ripple-carry adder, employing alternating positive and negative logic internally in the standard CMOS design saves one inverter for every two bit positions (Figures 17 and 19). Coupled with the logic savings, the symmetry of the design allows for tight, symmetric cell spacing with no dead space (unused area) in the logic block. As a result, both BIVOS designs suffer a significant area penalty due to the extra logic required for level conversion. For inverter level conversion, employing positive logic adds an inverter at each row and requires an additional $1.5\mu m$ between each row for p-well spacing at voltage boundaries. The traditional level conversion design only adds a single inverter and a single

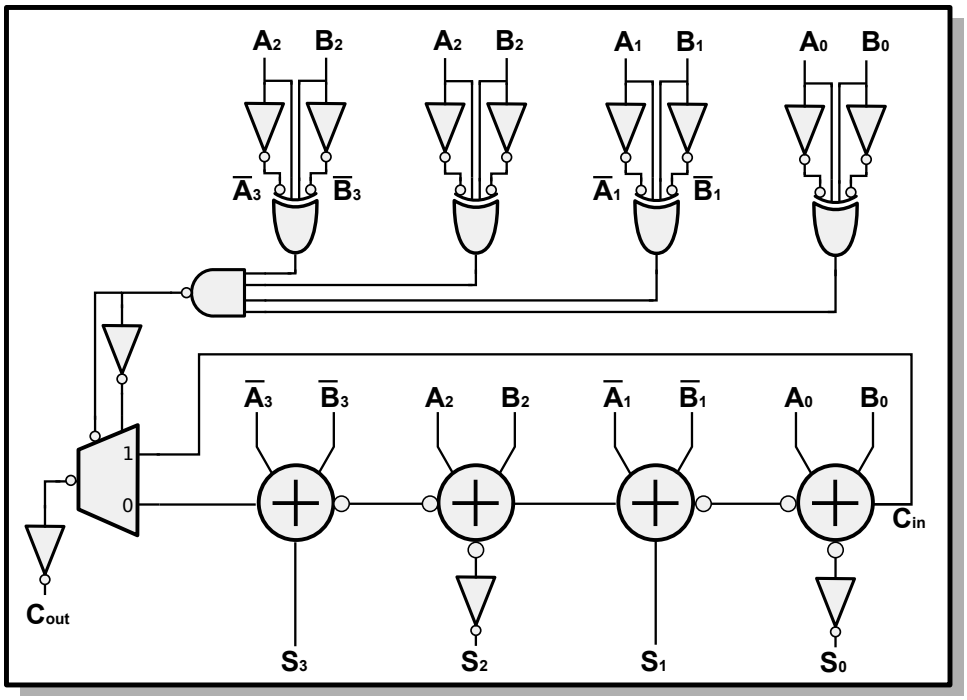


Figure 21: Standard CMOS implementation for a block-propagate adder. Utilizing alternating logic, the standard design eliminates three inverters at every two bit positions when compared to a positive logic design.

level converter, however, the width is equivalent to the inverter design due to unused dead space. Where traditional level conversion saves area is the need to add $1.5\mu m$ for p-well spacing between only two of the four rows. The result is a 34% and 18% area penalty for the inverter and traditional level conversion designs respectively.

Shown in Figures 21 and 22, the standard CMOS block-propagate adder saves three inverters for every two bit positions by capitalizing on the availability of both positive and negative input signals generated for the propagate logic. Despite these savings, the added complexity of the device requires additional logic that reduces the relative of impact of the removed inverters. The area penalty for the BIVOS designs, in turn, is not as pronounced as was the case for the ripple-carry adder. Requiring an additional six inverters and p-well spacing between each row, inverter level conversion yields a 23% area penalty. Needing only a single inverter and a single level converter along with p-well spacing between a pair of rows, traditional level conversion yields a 7% area penalty.

Finally, the standard CMOS carry-select adder saves only one inverter for every two

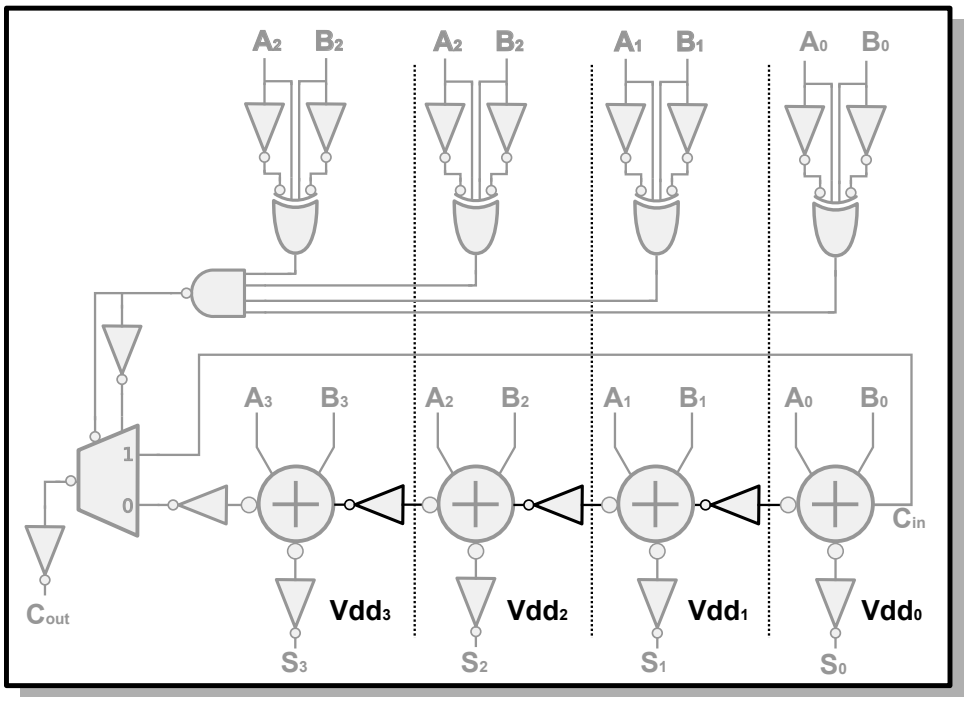


Figure 22: BIVOS block-propagate adder utilizing inverter-based level conversion. Implemented with positive logic, the design utilizes inverters as "level converters" between voltage boundaries. As was the case for the ripple-carry adder, the design is unable to utilize alternating logic.

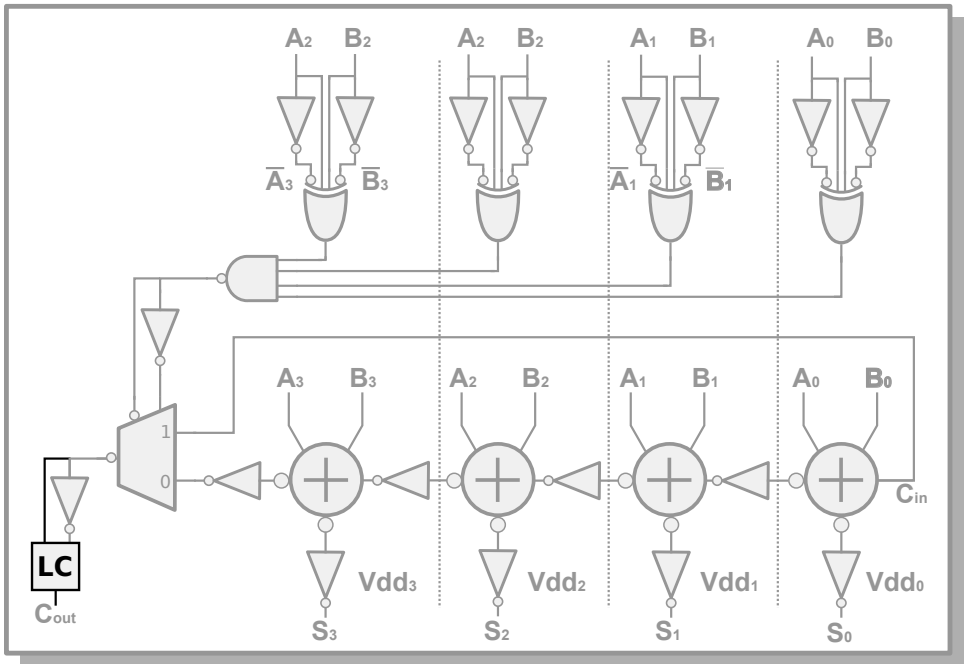


Figure 23: BIVOS block-propagate adder utilizing traditional level conversion. Implemented with alternating logic, the design requires only a single level converter between voltage boundaries. Limiting the number of voltage bins again allows for the application of alternating logic.

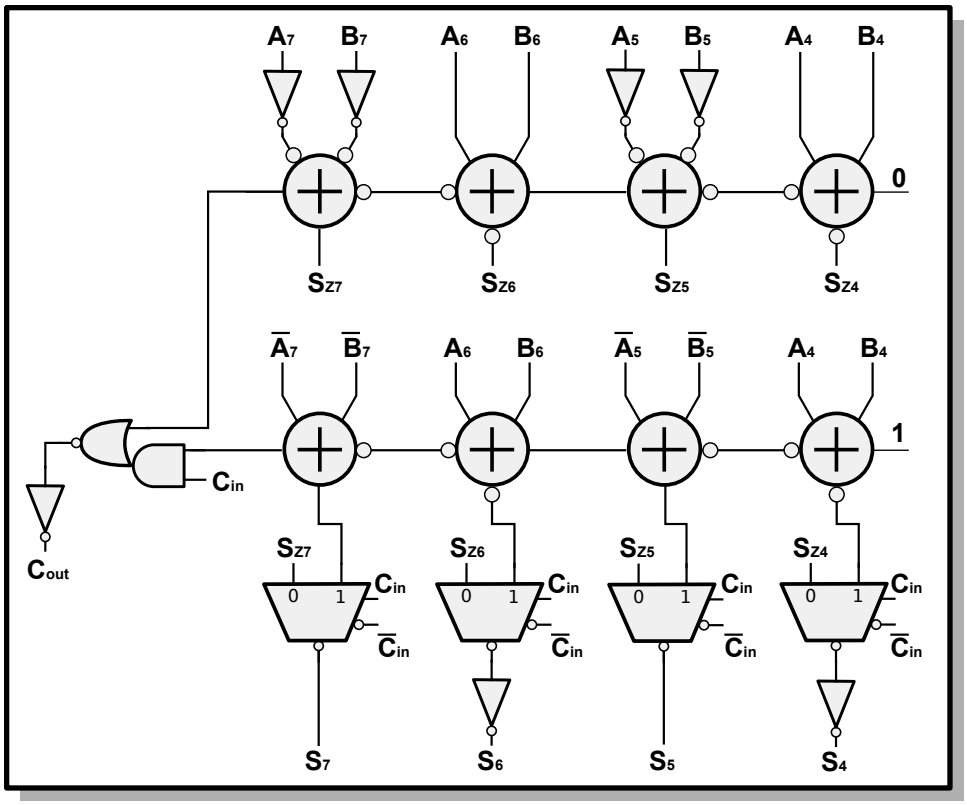


Figure 24: Standard CMOS implementation for a carry-select adder. Utilizing alternating logic, the standard design eliminates one inverter at every two bit positions when compared to a positive logic design.

bit positions by implementing alternating logic (Figure 24). At the same time, the design requires a subset of the addition logic to be duplicated with added multiplexers to resolve carry selection. Because of this irregular design, the resulting circuit footprint contains dead space that allows both BIVOS designs to insert the logic needed for level conversion with minimal area impact. In the case of inverter level conversion (Figure 26), the four added inverters and p-well spacing result in an area penalty of 18%. For traditional level conversion, the extra level converter and p-well spacing yields an area penalty of just 4%.

Beyond the additional overhead added for voltage conversion, the vertical routing of V_{dd} lines requires area. Here, design of a single datapath allows ample space in metal three for routing. Assuming this the case, the additional V_{dd} lines required to route multiple supplies results in no area penalty. Should the additional V_{dd} lines displace other signals, however, these routing requirements will add area to design overhead. With a rail width of $0.9\mu m$,

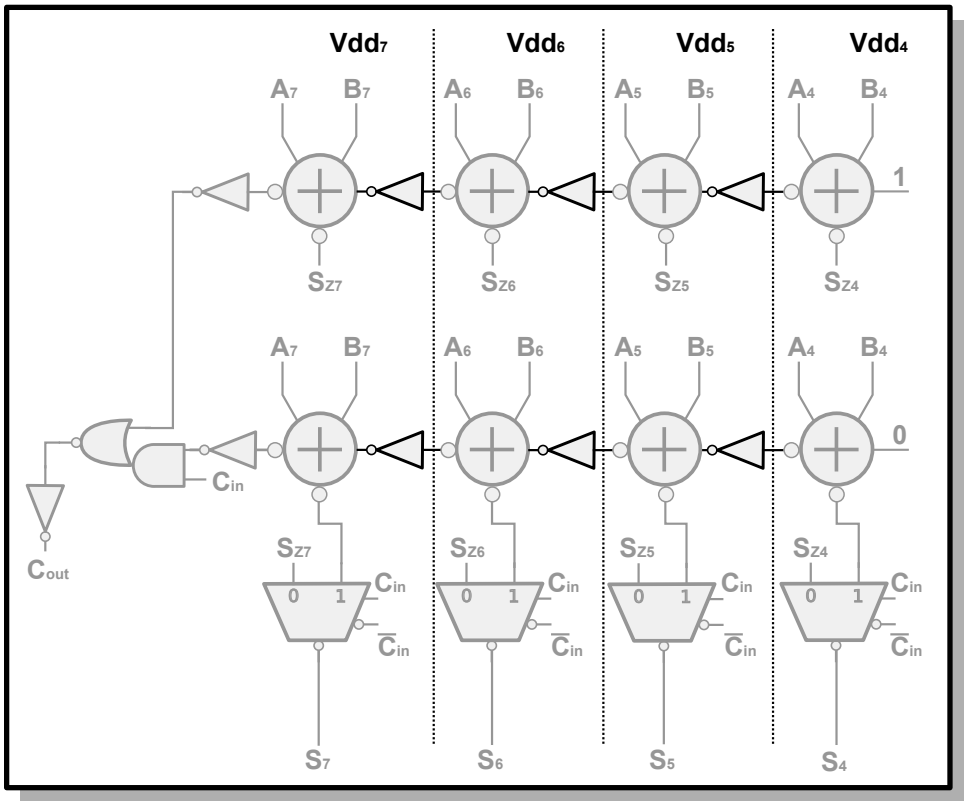


Figure 25: BIVOS carry-select adder utilizing inverter-based level conversion. Implemented with positive logic, the design employs inverters as "level converters" between voltage boundaries. Once again, the inclusion of inverters along the carry chain makes the use of alternating logic impractical.

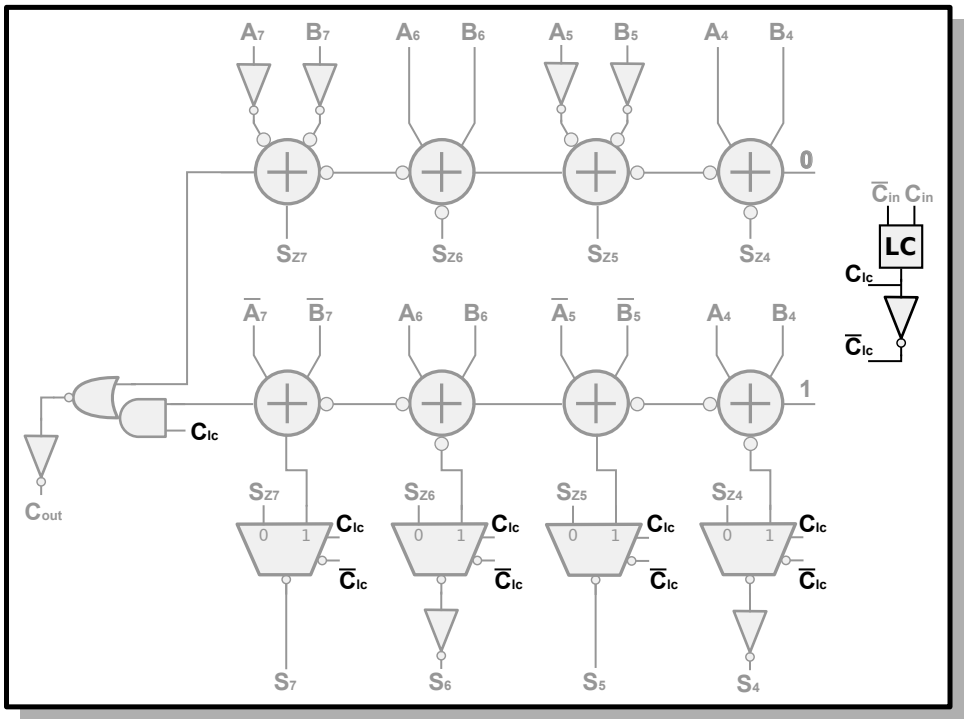


Figure 26: BIVOS carry-select adder utilizing traditional level conversion. Implemented with alternating logic, the design requires only a single level converter and a single inverter between voltage boundaries. With a limited number of voltage bins, transistor count is reduced through the use of alternating logic.

the potential area impact for each datapath (including metal-to-metal spacing of $0.3\mu m$) is approximately 2% per voltage level (V_{dd} width $\times$ Cell Height / Cell Area) routed in each architecture. Assigning an independent voltage to each bit then implies roughly a 14% area penalty (14 additional V_{dd} lines $\times$ Penalty).

Additional routing requirements are necessary to deliver the multiple voltage sources to BIVOS datapaths. If at all possible, BIVOS datapaths should be placed directly adjacent to voltage sources (whether on die or off). When this is not possible each voltage line requires equivalent area and, excluding any routing complications created by the additional lines, the expected impact is equal to the number of independent voltage levels multiplied by the routing expense for a single voltage line.

4.4 Simulation Methodology and HSpice Characterization

Simulation was employed to compare the effectiveness of BIVOS and reduced-precision solutions. While Spice is widely considered to be one of the most accurate circuit-simulation tools available, the accuracy that Spice delivers comes at the expense of simulation speed. Given the size of the circuits to be tested, full-scale Spice simulation was not possible. Instead, a custom C++ simulator was designed to substantially decrease simulation time. The simulator work flow, shown in Figure 27, breaks circuits into smaller pieces that are more manageable for Spice simulation. Layout is then utilized to estimate parasitics for these sub-circuits, followed by Spice simulation to create a device model and finally C++ simulation to emulate PCMOS behavior (Figure 27).

Under this methodology, HSpice simulation acts as a basis for the C++ simulator. To improve Spice simulation accuracy, layout was first performed for sub-circuit elements to ensure any modeling includes internal line capacitance. While the size of a sub-circuit is arbitrary in terms of the simulation methodology, the standard cells outlined in Section 4.1 were used here. With sub-circuit layout complete for the library of TSMC $0.18\mu m$ standard cells, a Spice netlist was extracted for each sub-circuit that included device parasitics. The resulting netlists were then combined with TSMC $0.18\mu m$ device models and characterized through HSpice simulation.

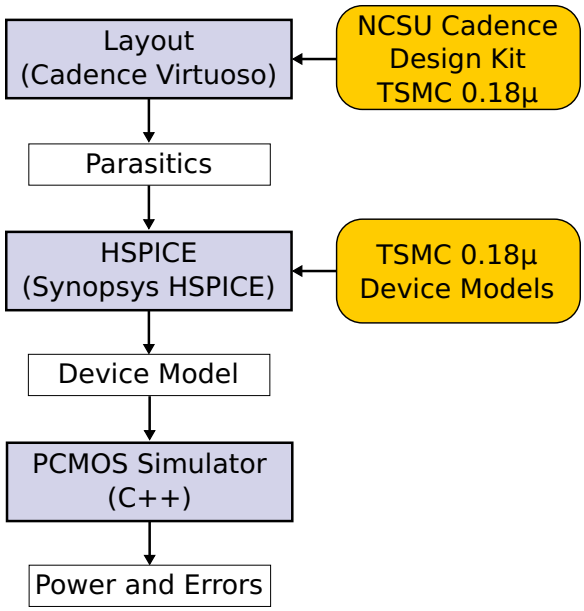


Figure 27: Workflow for the simulation methodology. Parasitics are first extracted by device layout. A device model is then created through Spice simulation. Finally, power and error estimates are accomplished through a custom PCMOS simulator.

During HSpice simulation, the sub-circuit outputs were loaded to match the input capacitance of the sub-circuits they are intended to drive. In addition, inverter pairs were added to drive sub-circuit inputs to better model the drive strength a cell would experience in circuit. Sub-circuits were then simulated to characterize power consumption and propagation delay at each output. Static power consumption was measured by transitioning circuit inputs over all possible input combinations and allowing the power draw to settle over $10\mu s$. Dynamic energy consumption was then measured by transitioning sub-circuit inputs over all possible state transitions and measuring average power consumption for $10ns$. Total and static energy consumption were first calculated as

$$Energy = Power \times Time, \tag{17}$$

then "dynamic" energy consumption was calculated as a combination of short-circuit and dynamic energy by subtracting out static energy consumption (Equation 18). Propagation delay for sub-circuit outputs was measured by transitioning sub-circuit inputs over all input state transitions resulting in output transitions. Delay was then measured from the time the trigger input reached $V_{dd}/2$ to the time when the transitioning output reached $V_{dd}/2$.

Simulations were repeated over a range of $0.8V$ to $1.8V$ supply voltages in $0.1V$ increments with input signal voltage also varied over $0.8V$ to $1.8V$ in $0.1V$ increments.

$$E_{total} = E_{dynamic} + E_{short} + E_{static} \tag{18}$$

To simulate a relative increase in thermal noise at a future technology generation, additive noise was injected by coupling a noise source to circuit inputs as reported in [20]. The injected thermal noise was modeled as a Gaussian random source with a standard deviation σ, referred to as the noise root-mean-square (RMS) value [40, 76]. Noise RMS was chosen to violate the minimal noise margin required to maintain deterministic circuit operation, defined for a modern process in Equation 19 where $\sigma = \sqrt{\frac{kT}{C}}$ [36].

$$V_{dd}/\sigma \geq 20 \tag{19}$$

$$\text{or} \qquad \sigma \leq 0.05 \times V_{dd} \tag{20}$$

For a TSMC $0.18\mu m$ process where nominal V_{dd} equals $1.8V$, thermal errors began to emerge as σ approached $150mV$, or $\sigma \approx 0.08 \times V_{dd}$. With $\sigma = 150mV$, reductions in supply voltage resulted in probabilistic device operation. Increasing noise levels beyond this point, where $\sigma > 150mV$, would render the device probabilistic even at the nominal operating voltage. As such, $\sigma = 150mV$ was chosen to model the point where conventional voltage scaling would be limited by the bit errors caused by thermal noise.

The full-adder sub-circuit was chosen as the noise injection point for thermal noise testing. All other sub-circuits had substantially fewer transistors than the full adder and were considered circuit extensions for the purpose of noise modeling. Thermal noise was injected into the HSpice netlist in the form of three Gaussian piece-wise-linear sources. Each noise source was coupled to a circuit input causing random fluctuations in the signal voltage. By coupling noise sources to full-adder inputs, as opposed to outputs, the resulting characterization captured any error masking performed by the full adder (discussed in [7]). Input signals were held constant and actual and expected circuit outputs were compared to

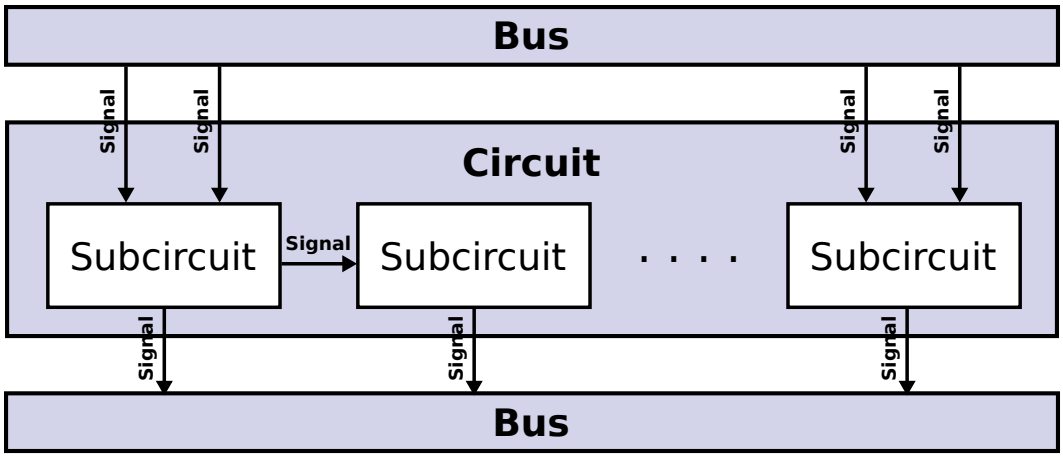


Figure 28: Subcircuits, as modeled through Spice simulation, form the base element for the PCMOS simulator. Signals then connect various subcircuits to form larger circuits of arbitrary size.

determine the probability of correctness p as

$$p = \frac{\text{correct} \quad \text{samples}}{\text{total} \quad \text{samples}}. \tag{21}$$

Probability simulations were repeated once for every possible input combination over $10,000$ clock cycles at $300ns$.

4.5 A Custom PCMOS Simulator

Based on sub-circuit device models derived through HSpice, a C++ PCMOS simulator allows for circuits of arbitrary size. Each circuit is comprised of individual sub-circuits (Figure 28) and each sub-circuit acts as a black box based on model data—accepting inputs and accounting for delay, energy, and injecting any bit-errors at appropriate rates to calculate outputs. Sub-circuits are "wired" to form circuits with sub-circuit outputs driving the inputs of other sub-circuits. As inputs are applied to the circuit they are allowed to propagate through the various sub-circuits where each calculates both an ideal and a PCMOS solution. Circuit outputs are then sampled and the PCMOS-bit solutions are compared to the ideal-bit solutions to determine error rates. Once simulation has completed for the given number of circuit samples, error counts are compared to determine bit-level error rates and mean squared error (MSE) is calculated to determine the average magnitude deviation.

Internally, sub-circuits use gate level analysis that is based on event-driven logic simulation [8, 26, 72, 99]. Under this event-driven model, events are defined as logic transitions ($1 \rightarrow 0$ or $0 \rightarrow 1$) on circuit signals. Each circuit signal maintains a list of events that includes the event time, relative to the current clock, and signal value (0 or 1). When a sub-circuit is clocked, the event lists from all sub-circuit inputs are aggregated and sorted for time of arrival. Sub-circuit outputs are then calculated for each event in the master event list. HSpice model data is used to calculate dynamic energy consumption and output delay for each event based on the current and previous input states. Dynamic energy is added to total energy consumption and the calculated propagation delay is added to event times and placed on output signals as new signal events. Should any event arrive before a prior event can fully propagate, the pair are merged and recalculated as a single event based on the time of arrival of the latest event. Once the master event list has been exhausted, static energy consumption is calculated and added to total energy consumption based on the final sub-circuit input state and clock rate. Once signal events have propagated through the entire circuit and calculations are complete, all signal events are cleared and a new clock cycle begins.

Validation of the PCMOS simulator was performed by comparing energy consumption results to HSpice simulation for an 8-bit, ripple-carry adder (Figures 19 and 90). The previous layout was modified to isolate the power planes for each cell within the circuit to determine the energy consumption at each cell. A Spice netlist was then extracted from the adder layout to include internal line capacitance. Thermal noise was then injected at A and B full-adder inputs (C inputs were isolated within the circuit) using a Gaussian random distribution as before. Adder inputs were driven using a uniform random distribution and the circuit was sampled over $10k$ samples to determine power consumption. This was then compared to an equivalent 8-bit, ripple-carry adder implemented in the C++ simulator and simulated over $10k$ samples.

The experiment was repeated over four biasing configurations, shown in Table 4, with no noise and again with $150mV$ RMS thermal noise. The four configurations chosen represent nominal voltage, minimal voltage, a BIVOS distribution with $0.1V$ increments between

Table 4: Biasing Configurations Employed for PCMOS Simulator Validation

Bias	**Bit 7** (V)	**Bit 6** (V)	**Bit 5** (V)	**Bit 4** (V)	**Bit 3** (V)	**Bit 2** (V)	**Bit 1** (V)	**Bit 0** (V)
1	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
2	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
3	1.8	1.8	1.7	1.6	1.5	1.4	1.3	1.2
4	1.8	1.8	1.3	1.0	0.9	0.8	0.8	0.8

Table 5: Energy Consumption per Clock Step for an 8-bit Ripple-Carry Adder

Bias	**RMS** (mV)	**HSpice** (fJ/Clock)	**C++** (fJ/Clock)	**Error** (%)
1	0	721	664	7.90
2	0	136	141	3.75
3	0	569	551	2.94
4	0	573	574	0.29
1	150	740	702	4.96
2	150	140	142	1.70
3	150	574	571	0.40
4	150	582	588	1.09

voltage planes, and a BIVOS distribution with increments up to $0.5V$ between voltage planes. The two BIVOS distributions were chosen specifically to test low (3) and high (4) static current flows between voltage planes.

Each of the biasing configurations tested for the PCMOS simulator matched the HSpice validation results within tolerable limits (Table 5). The worst case error was just under 8% for the nominal voltage configuration with no noise present. In particular, the two BIVOS configurations testing high and low static current draw with thermal noise matched closely with a deviation on the order of 1%.

Further inspection of the energy consumption for each of the individual full adders within the ripple-carry adder also shows a close alignment between the PCMOS simulator and HSpice results for the majority of the configurations. Shown in Figure 29, the estimated energy consumptions match to within 20% of the HSpice validation results in all cases. Evident from the mismatch at the lower bit positions for Bias 1 (blue lines), the algorithm

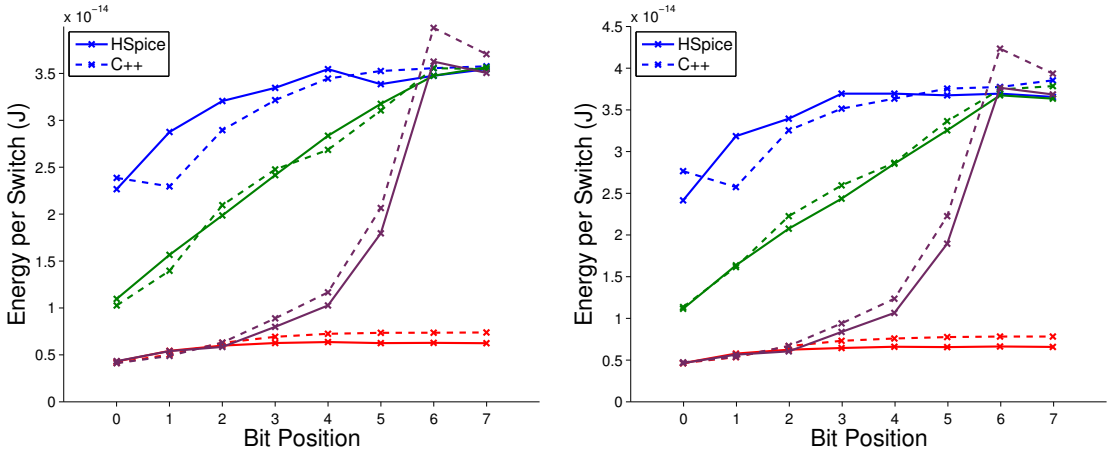


Figure 29: Energy consumption of full adders, by bit position, for the four bias configurations detailed in Table 4 as measured by HSpice and PCMOS simulation: (a) $0V$ noise RMS and (b) $150mV$ noise RMS. The PCMOS simulator and HSpice results align well for the most part; the largest deviation (roughly 20%) occurs with an underestimation of energy consumption for the full adders in the nominal configuration ($1.8V$).

used for merging signal events does not perform well for signals with close arrival times. As a result, the minimal propagation delay for $1.8V$ biasing coupled with a short carry chain at the lower-order bit positions allows the algorithm to merge signals over aggressively. This yields low power consumption estimates for nominal configurations. The full-adder results do, however, show that the PCMOS simulator closely models increasing energy consumption with increasing bit positions due to delay along the carry chain causing spurious switching as signals propagate. Evident from Table 5, any deviations at individual full adders are averaged out at the aggregate with the largest errors occurring as under estimates for baseline energy consumption (the two nominal configurations).

The PCMOS simulator was additionally validated for delay measurements by comparing worst-case propagation results to HSpice simulation for the same 8-bit, ripple-carry adder used to verify power measurements. In both cases, one ripple-carry input was held at negative one and the second input was transitioned from zero to one to activate the entire carry chain. Propagation delay was measured from the time the zero-to-one transition reached $V_{dd}/2$ to the time that the most significant output bit reached $V_{dd}/2$. The experiment was

Table 6: Comparison of HSpice and PCMOS Simulator Measurements for Propagation Delay

Voltage	**HSpice** (nS)	**C++** (nS)	**Error** (%)
1.8	1.35699	1.3962	2.81
1.7	1.44066	1.4872	3.13
1.6	1.53728	1.6044	4.18
1.5	1.66022	1.7453	4.87
1.4	1.80883	1.9298	6.27
1.3	2.00342	2.1715	7.74
1.2	2.24417	2.5061	10.45
1.1	2.60844	2.9948	12.90
1.0	3.12856	3.7447	16.45
0.9	3.94258	4.9989	21.13
0.8	5.45996	7.4893	27.10

repeated once for uniform voltage distributions ranging from $1.8V$ to $0.8V$ in $0.1V$ increments and again for the same biasing configurations in Table 4. In all cases noise RMS was fixed at $0.0V$.

Shown in Table 6, delay measurements for uniform distributions match closely at nominal voltage levels with a deviation of less than 3%. As voltage scaling increases, however, PCMOS simulator results deviate from HSpice results with each voltage step. Supply voltages at $1.2V$ and above match within 10% of HSpice measurements. Below $1.2V$, PCMOS simulator results begin to diverge fairly substantially with $0.8V$ measurements exhibiting a 27% deviation from HSpice measurements.

In the case of the four biasing configurations from Table 4, propagation delay measurements again show increased deviation with increased voltage scaling. Shown in Table 7, the measurement error for a BIVOS distribution is determined primarily by the lowest supply voltage employed. Biasing configurations 2 and 4 both employ $0.8V$ supplies and the deviation of each roughly equals the 27% deviation exhibited by the $1.8V$ uniform distribution. Similarly, bias 3 employs $1.2V$ and the deviation is roughly equal to the 10% error exhibited by the $1.2V$ uniform distribution. As a result, the deviation in delay measurements for a particular biasing scheme can be approximated by the error exhibited for a

Table 7: Worst Case Propagation Delay for an 8-bit Ripple-Carry Adder

Bias	**HSpice** (nS)	**C++** (nS)	**Error** (%)
1	1.35699	1.3962	2.81
2	5.45996	7.4774	26.98
3	1.66745	1.8117	7.96
4	3.4026	4.5927	25.91

uniform distribution at the lowest supply voltage employed. While these measurements can be fairly inaccurate at lower supply voltages, the result can be bounded with expected error and they do provide a reasonable estimate to compare the performance impact of various distributions.

Error propagation within the PCMOS simulator was validated against HSpice simulation using a chain of eight full-adders configured in a ripple-carry pattern. Under typical conditions, the entire circuit would be exposed to thermal noise. As outlined in 4.4, however, thermal noise was modeled as Gaussian noise sources at each of the full-adder inputs with additional elements acting as extensions to the full adder. Accordingly, Inverters (necessary for a standard ripple-carry adder) were excluded to remove any filtering effects they would add to the HSpice simulation.

Simulation was performed over $10m$ data points at $150mV$ noise RMS. A comparison of the resulting probabilities of correctness, as measured at circuit outputs, is shown in Figure 30. PCMOS simulator measurements match to within 0.1% of the corresponding HSpice measurements at all bit positions. Further, error distributions (or correctness distributions as show) largely follow the same pattern in each case, indicating that error propagation and filtering is properly addressed within the PCMOS simulator. The PCMOS simulator results do slightly under estimate probability of correctness in most cases, although the discrepancy is minor.

4.6 Estimating Mean-Squared Error

Shown in Figure 5, the probability of a correct computation at $0.15V$ noise RMS is exceptionally high. In turn, the probability of an error is exceptionally low (on the order of 1

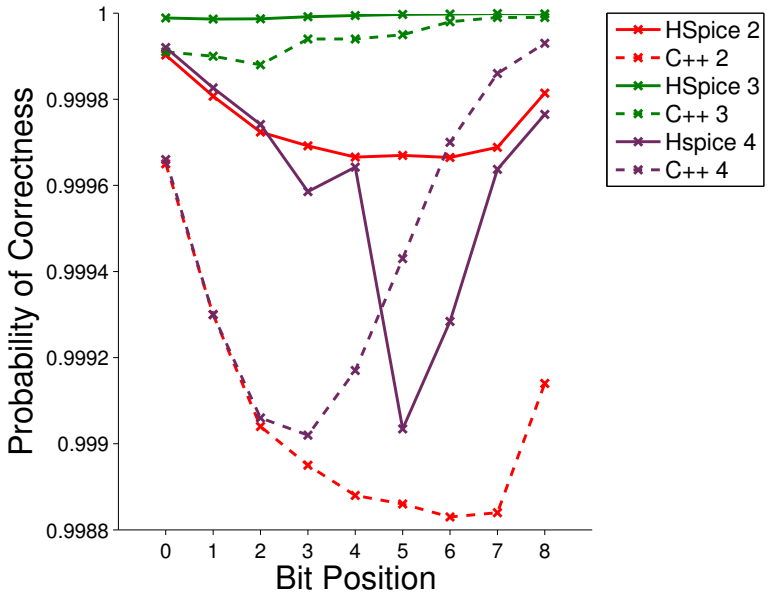


Figure 30: Probability of correctness, by bit position, for bias configurations two, three, and four (detailed in Table 4) as measured by HSpice and PCMOS simulation at $150mV$ noise RMS. The PCMOS simulator results largely follow the pattern of HSpice results, although at a slightly reduced probability of correctness.

in $10k$ samples). Because of these low error rates, the Monte Carlo simulation technique utilized in the PCMOS simulator requires a high number of samples to realize the true operating probability, or the associated MSE, of a circuit (roughly $10m$ samples). While the simulator is substantially faster than Spice simulation and works well as a PCMOS emulator, it requires multiple days to process the high number of samples necessary to realize an accurate MSE and is not nearly fast enough for rapid design-space exploration.

Instead, design-space exploration is accomplished by using the PCMOS simulator to determine power requirements (validated as accurate at $10k$ samples in Section 4.5) in conjunction with mathematical analysis to determine theoretical MSEs. Previous work in estimating error rates focused on bit errors at circuit outputs [7, 27, 45, 46, 47]. Three works in particular [45, 46, 47], develop mathematical models that greatly improve simulation time while closely matching error-rates derived through Spice simulation. While these techniques perform well and provide a means to compare error-prone designs, they offer no way to derive MSE as output error probabilities provide no indication of error magnitude (Figure 11). Rather than rely on output bit-error rates, a Matlab script was designed to estimate MSE

by iterating over error combinations based on Equation 7. Circuits are subdivided into full adder operations with sum outputs weighted by bit position i and carry outputs weighted by $i + 1$. Identical to the PCMOS simulator, the probability of correctness p_i for each full adder operation is determined by subcircuit-device models and biasing voltages.

As subcircuit-device models specify p by input combination, each circuit is simulated in the PCMOS simulator over $100k$ samples (again uniform random input distributions) to determine the input combination probabilities at each full adder. Circuits are operated with no errors to establish input probabilities under deterministic operation. The resulting characterization allows the probability of correctness to be calculated at each full adder output based on individual adder position and biasing voltage.

Deviating from Equation 7, iterating over all possible error combinations quickly becomes prohibitive as the number of full adders grows. Instead, the expected MSE is estimated by only considering the most significant contributors. The script starts at the most significant bit position (as they contribute more to a solution), calculating error magnitude and probability of occurrence for a single bit error, and stores the error magnitude as the maximum error contribution. It then recurses calculating the error magnitude and probability of occurrence for the original bit error plus a second, simultaneous bit error. The resulting contribution is compared to the maximum contribution, replacing the maximum contribution if exceeding it. Recursion repeats, adding additional errors, until the error contribution becomes insignificant (less than 0.001% of the maximum contribution). Once the initial bit error iterates through all significant combinations of errors at lower bit positions, the script moves onto the next most significant bit position and recursion begins again. This is repeated as long the resulting contributions are greater than 0.001% of the maximum contribution.

Validation of this estimation technique was performed by comparing results from the PCMOS simulator to estimated MSEs for a 16-bit, ripple-carry and an 8-bit, array multiplier. Starting with a fixed probability of correctness at $p_0 = 0.99$ (all other $p_i = 1$), each device was simulated over $100k$ samples in the PCMOS simulator and the estimated MSE was derived—probability of correctness was fixed to 99% to ensure the PCMOS simulator

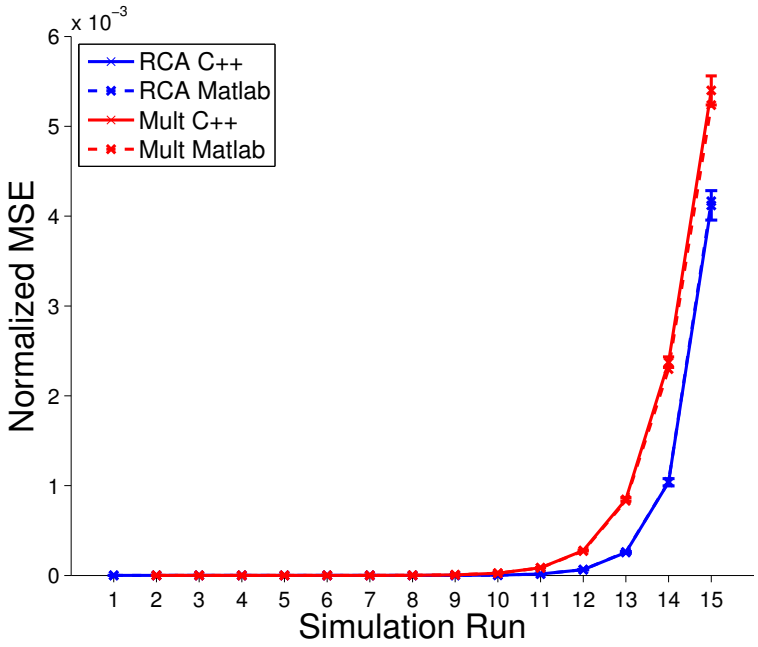


Figure 31: A comparison of estimated and simulated MSEs for 15 different biasing configurations. Estimated error closely aligns with simulated results for all simulation runs with deviations on the order of 1%. Ninety-five percent confidence intervals, calculated for PCMOS simulator results, are largely imperceptible for all but the largest MSEs (implying a high degree of confidence in the MSE point estimates).

could realize the true operating probability of each circuit with $100k$ samples. This was repeated over 15 trials by adding an additional, adjacent bit at $p_i = 0.99$ for each trial to test accuracy of estimation at each bit position. Shown in Figure 31, the resulting estimated MSEs closely align with simulated results, deviating by no more than 3%. Further, 95% confidence intervals (calculated for PCMOS simulator results) fall within $\pm 5\%$ of MSE results for all configurations tested.

Finally, a single experiment was completed using subcircuit-device models to evaluate more realistic parameters. Input voltages started at $1.8V$ at bit 15 and were reduced at each adjacent bit by $0.1V$ down to $0.8V$ at bits 0 through 5. Each device was simulated over $10m$ samples in the PCMOS simulator to ensure the true operating probability of each circuit was realized. Shown in Table 8, the resulting estimated MSEs closely match those derived by simulation and 95% confidence intervals fall within $\pm 10\%$ of measure results.

Table 8: Comparison of MSE Results

	PCMOS Simulator	**Estimated MSE**	**Variation**	**Confidence**
RCA	$1.78E+004$	$1.75E+004$	1.52%	$\pm 1.88E+003$
Mult	$4.63E+004$	$4.60E+004$	0.58%	$\pm 3.43E+003$

4.7 Comparing a Selection of Circuits

Three fixed-point circuits were chosen for a comparison between BIVOS and reduced precision solutions: a ripple-carry adder, an array multiplier, and an FIR filter. Reduced precision solutions were achieved through power down of unused portions of the circuit. This is roughly equivalent to a reduced precision hardware implementation for the ripple-carry adder. For the array multiplier, however, inputs and outputs were hard wired to the outermost full adders along one dimension of the two dimensional structure. As a result, it was only possible to power down one dimension of the structure.

Circuits were sized for a width to allow for 16 configurable bit positions. Eight reduced precision solutions were then tested for each circuit with widths varied from 15 down to 8-bits. Eight BIVOS configurations were also tested for each BIVOS circuit design. For designs employing inverter-based level conversion, an initial bias position, with a nominal voltage of $1.8V$, was varied from bit 15 down to bit 8 and the supply voltage of each adjacent bit was reduced by $0.1V$. For designs employing traditional level conversion, bits were divided into two voltage bins where the most-significant bin received $1.8V$ and the least-significant bin was varied in $0.1V$ increments between $1.7V$ and $1.0V$ (repeated for low-order bin sizes of 10, 11, 12, and 13 bits). In all cases, simulation results were compared to ideal, full-width results to determine MSE. Input signals were randomly generated using uniform random distributions (fully exercising the number range) and all experiments were executed with a noise RMS of $150mV$ over $10k$ circuit samples at a clock period of $300ns$.

4.7.1 Ripple-Carry Adder

The ripple-carry adder, comprised of a single dimension of full adders with one at each bit position, had the simplest circuit structure of the three circuits tested. Based on the

requirement of 16 configurable bit positions, it was designed for an input width of 16-bits with a 17-bit output. As discussed in Section 4.3, the nominal and BIVOS designs employing traditional level conversion utilized alternating logic to minimize transistor counts. BIVOS designs employing inverter level conversion were forced to use positive logic.

Shown in Figure 32, the full-adder switching activity is considered for a standard CMOS circuit along with three aggressive power reduction solutions: reduced precision with 8 bits powered down, BIVOS employing inverter level conversion with an initial bias position at bit 15, and BIVOS employing traditional level conversion with 13 bits biased to $1.0V$. Switching activity in each of the full-adders indicates power consumption relative to circuit architecture. At nominal operation, the standard CMOS adder exhibits low circuit activity for bit positions one and two before increasing into an alternating high-to-medium pattern (Figure 32 (a)). Low activity at the low order bit positions is a function of a fixed carry-in of zero at bit 0 and is present for all adder configurations. With each full adder along the carry chain, however, the probability of a carry-in increases due to the uniform input distribution. By bit 4, the probability of a carry-in approaches 50%. Once the probability of a carry-in is sufficiently high, full adders are likey to switch twice: once when circuit inputs arrive and again once the carry-in has had an opportunity to propagate from lower order bit positions. By eliminating inverters along the carry chain and inserting inverters at negative-logic positions in the alternating-logic adder, input delay is increased at these negative-logic postions while carry-in delay is decreased. This allows the negative-logic full adders extra opportunity to merge input and carry-in signals resulting in decreased switching activity at alternating bit positions. While the reduced-precision solution has no activity at the powered-down bit positions, shown as white boxes, the powered bit positions show a pattern that is similar to the standard CMOS solution (Figure 32 (b)).

Contrary to the standard CMOS design, the BIVOS solution employing inverter level conversion shows the highest circuit activity from bit positions 5 through 10 (Figure 32 (c)). This is a result of circuit propagation occurring in a series of waves due to biasing configuration. In a typical multi-voltage design, supply voltages are assigned to minimize critical path propagation and ensure all paths complete at roughly the same time. As a

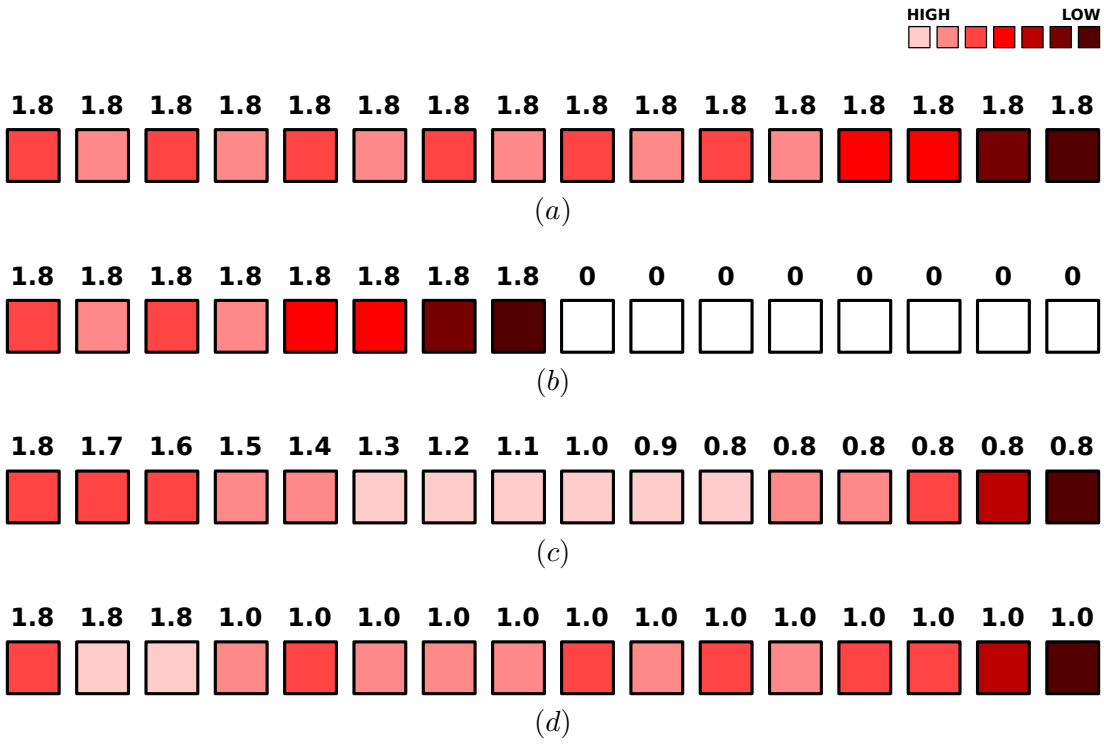
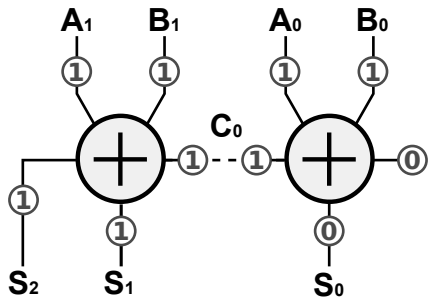


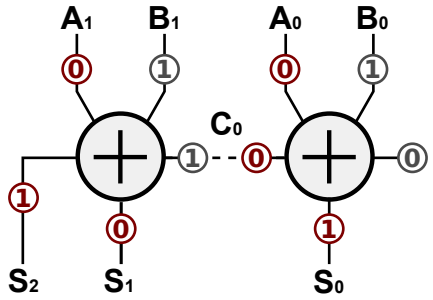
Figure 32: Switching activity for a 16-bit, fixed-point, ripple-carry adder: (a) Nominal operation at full ($1.8V$) voltage, (b) Reduced precision operation with eight bits powered down, (c) BIVOS operation with an initial bias position at bit 15, and (d) BIVOS operation with 13 bits biased to $1.0V$. The delay introduced through voltage biasing in BIVOS operation creates propagation waves, causing additional circuit switching at higher-order bit positions. This translates to higher circuit activity at boundaries with low biasing voltages: through the middle of the circuit in (c) and at the high order bit positions in (d).

result, all voltages along a propagation path are equal ensuring a single propagation wave through the circuit. Conversely, a BIVOS solution ensures that low-voltage elements at low-order bit positions are driving high-voltage elements at high-order bit positions. This allows the high-order bit positions extra opportunity to complete before results from the slower, low-order bit positions are able to propagate. As a result, each biasing voltage creates an independent propagation wave that causes higher-order bit positions to recalculate, consuming extra power in the process. At voltages close to $1.8V$ the impact is small; at voltages close to $0.8V$, however, the added delay due to low-voltage operation can increase circuit activity significantly.

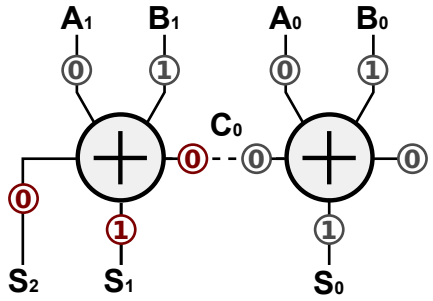
Example 12: To highlight the affect of propagation waves on switching activity, consider a positive-logic, two-bit, ripple-carry adder. Initially both A and B inputs are set to three resulting in a summation of six with a positive carry between the full adders.



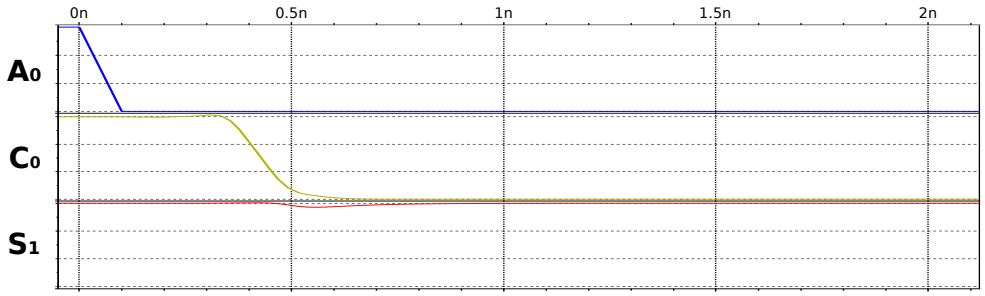
Switching A to a zero causes the full adders to recalculate. If the carry calculation at bit zero is unable to complete before the full adder at bit one completes, S_1 and S_2 will be calculated with the initial C_0 value.



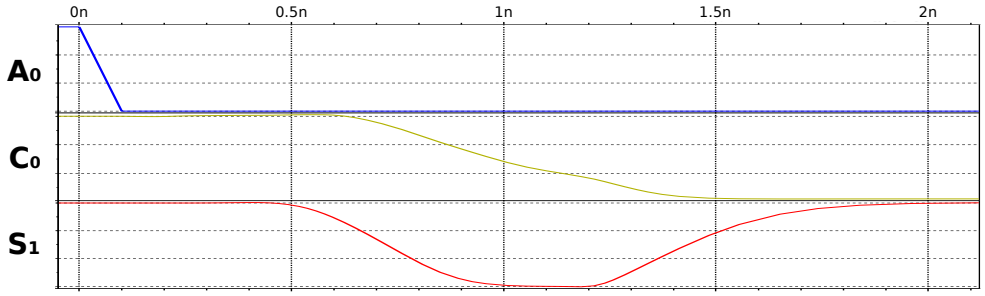
Once the carry at bit zero is calculated, the full adder at bit one will recalculate the S_1 and S_2 outputs for a second time. The delay incurred along the carry chain results in two switches at bit one while the inputs switched only once.



Under normal operating conditions both full adders receive $1.8V$. As the full adder is designed to minimize carry propagation delay (at the expense of sum propagation delay), this allows ample time for the carry output from bit zero to reach the carry input at bit one before the sum calculation can complete. Shown below for a full-adder pair simulated in HSpice, the S_1 output only momentarily glitches to $1.7V$ (well above threshold voltage) at $0.5ns$ due to the propagation delay at C_0.



With bit zero biased to $1.0V$, however, the delay at C_0 is increased due to the voltage scaling. As bit one is still operating at $1.8V$, S_1 completes well in advance of C_0 and must recalculate once C_0 arrives. Shown below for a second full-adder pair simulated in HSpice, the S_1 output fully switches to $0V$ before switching back to $1.8V$.



The result is an additional circuit switching due to the propagation waves created by voltage biasing, compared to nominal operation, for specific input patterns. □

From Figure 32 (c), the impact of the increased propagation delay due to $0.8V$ operation at bits 0 through 5 accumulates in a substantial increase in circuit activity in the middle of the device. As operating voltage increases by bit significance, the likelihood of increased activity propagating is absorbed due to the low probability of a fully activated carry chain.

At the high-order bit positions circuit activity returns to normal. The same affect is evident in the BIVOS solution employing traditional level conversion (Figure 32 (*d*)), however, the low-voltage bin extends all the way to bit 13 allowing increased circuit activity at the full-voltage, high-order bit positions. To some extent the impact of increased switching activity is mitigated in the inverter-based design by voltage scaling along the inner bit positions where increased activity occurs. In the case of the design employing traditional level conversion, increased switching activity is particularly expensive as this increases energy consumption at full adders that were already operating at full power.

Shown in Figure 33, the cumulative error count by bit position is examined for the same three aggressive power reduction solutions considered for switching activity. The reduced-precision solution, shown in red, allows a substantial number of bit errors, but constrains them to lower-order bit positions. The resulting MSE is then a function of the volume of bit errors occurring at these lower-order bit positions. Both BIVOS solutions, conversely, allow higher-order errors to occur and limit the overall error count (the solution employing inverter level conversion biases the bulk of the bit errors to low-order bit positions). In these cases, MSE is largely a function of the weight of the highest-order bit errors.

The resulting MSE versus average switching energy is shown in Figure 34 for the reduced-precision and inverter-based, BIVOS, ripple-carry-adder solutions. Both methods yield considerable energy savings compared to the baseline energy profile. The BIVOS implementation, however, yields energy savings with less MSE than a comparable reduced-precision implementation for a fixed energy budget.

Of note in Figure 34, the reduced precision solution exhibits a sawtooth pattern with many lower energy solutions resulting in lower a MSE than an adjacent higher energy solution. This is due to the alternating logic employed coupled with the uniform, random input distribution. With such a data set, all input combinations are equally likely at each full adder (Table 9). The expected value of an individual full adder is then 2.25—the average summation of all eight possible input combinations. When a positive-logic full adder is powered down, or reduced, the adjacent negative-logic full adder receives a fixed zero at the Cin input. The negative-logic full adder interprets this zero as a one. Coupled

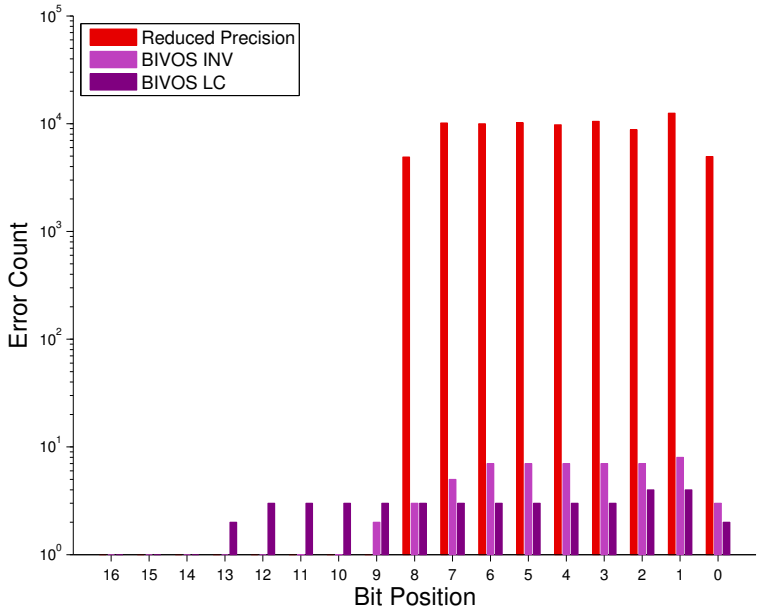


Figure 33: Bit-error rates by bit position for 3 ripple-carry-adder implementations: reduced precision operation with 8 bits powered down, BIVOS operation with an initial bias position at bit 15, and BIVOS operation with 13 bits biased to $1.0V$. Where a reduced-precision solution introduces a substantial number of errors at low-order bit positions, BIVOS solutions distribute a small number of errors across a larger range of bit positions.

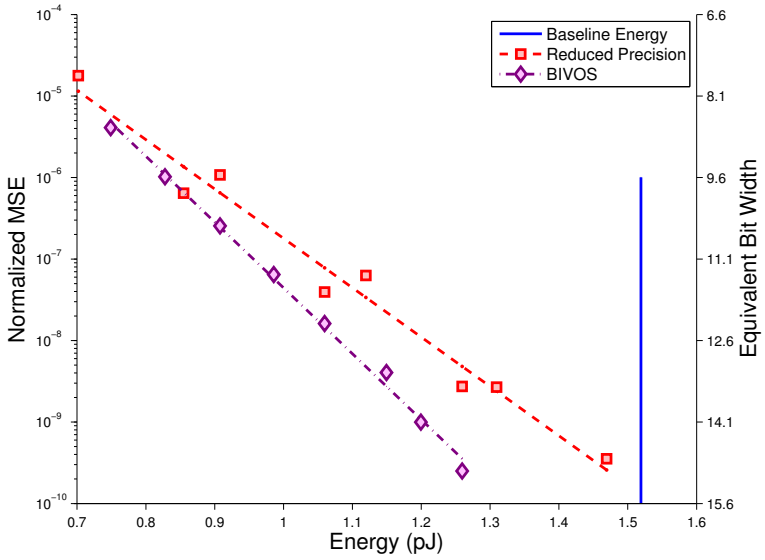


Figure 34: MSE vs energy for a fixed-point, ripple-carry adder. Both reduced precision and inverter-based BIVOS designs significantly reduce energy consumption when compared to a standard CMOS adder. The BIVOS solution, however, reduces error for any given energy level.

Table 9: Expected Full Adder Summation

A	B	Cin	Sum
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	1
1	0	1	2
1	1	0	2
1	1	1	3

with the S output on the positive-logic full adder fixed at zero, this translates to a constant summation of two at the powered down full-adder that is roughly equivalent to the expected value. Conversely, powering down a negative-logic full adder is interpreted as a zero at the adjacent positive-logic full adder resulting in a constant summation of zero that deviates substantially from the expected value. As a result, powering down a positive-logic full adder can offset some of the error introduced by an adjacent negative-logic full adder at high-order bit positions.

Repeated for the four BIVOS designs employing traditional level conversion, Figure 35 shows that designs based on traditional level conversion can also significantly reduce energy consumption. These designs, however, exhibit a trade-off between accuracy and propagation delay that is highlighted by the arching slope of the BIVOS solution spaces. Designs with fewer bias bits display better MSE characteristics, although they suffer from an increased number of bit positions that are susceptible to power increases due to propagation waves. As voltage is scaled power consumption increases at high-order bit positions resulting in a diminishing return per voltage step. When coupled with a consistent MSE penalty per voltage step, the culmination is a relative increase in MSE penalty per energy step. This affect is most prominent in the BIVOS design employing a 10-bit, low-order voltage bin.

Beyond the accuracy/delay tradeoff, not all BIVOS configurations outperformed a reduced-precision solution. Where the inverter-based BIVOS design allowed increasing supply voltages to be assigned by increasing bit significance, the designs employing traditional level

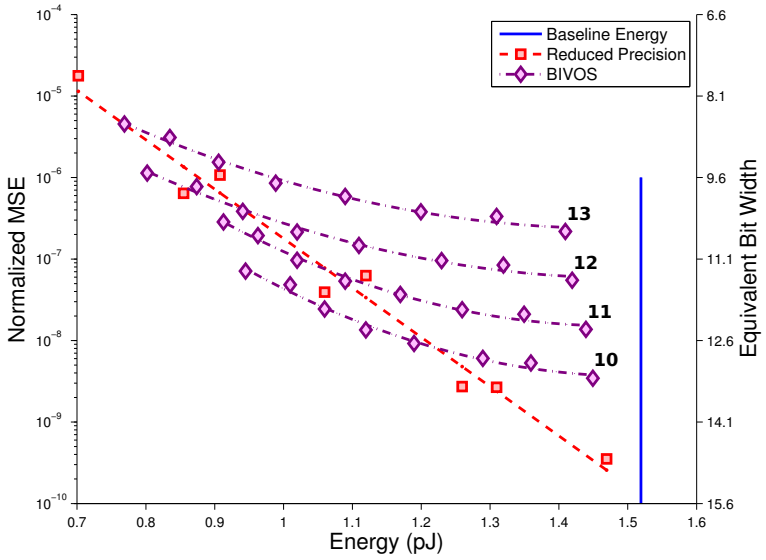


Figure 35: A comparison of fixed-point, ripple-carry adder designs employing standard CMOS, reduced precision, and BIVOS utilizing traditional level conversion. The four BIVOS designs were repeated with low-order voltage bins ranging from 10 to 13 bits. As was the case for the inverter-based BIVOS design, the energy savings are significant for all cases. The reduced precision solution, however, outperforms the BIVOS designs for most data points.

conversion only allowed for two distinct voltage levels. This provided a minimal opportunity to assign voltage levels by bit significance, requiring a compromise between the number of bit positions biased and the biasing voltage selected. Of the four bins sizes chosen, each of the resulting solution spaces exhibit a crossover point where a BIVOS solution becomes favorable to a reduced-precision design. Between the four designs, the solutions with fewer bits in the low-order bin displayed the most data points with favorable MSE.

The worst-case propagation delays for each of the solutions are shown in Figures 36 and 37. Where the reduced-precision solution decreases propagation delay, the voltage scaling employed with BIVOS solutions results in an increase in propagation delay. While all of the BIVOS solutions considered result in a substantial performance penalty, those employing traditional level conversion result in a slightly lower worst-case propagation delay than those employing inverter level conversion. To some extent this is a function of operating with higher supply voltages, but it is also a function of reduced circuit count through alternating logic.

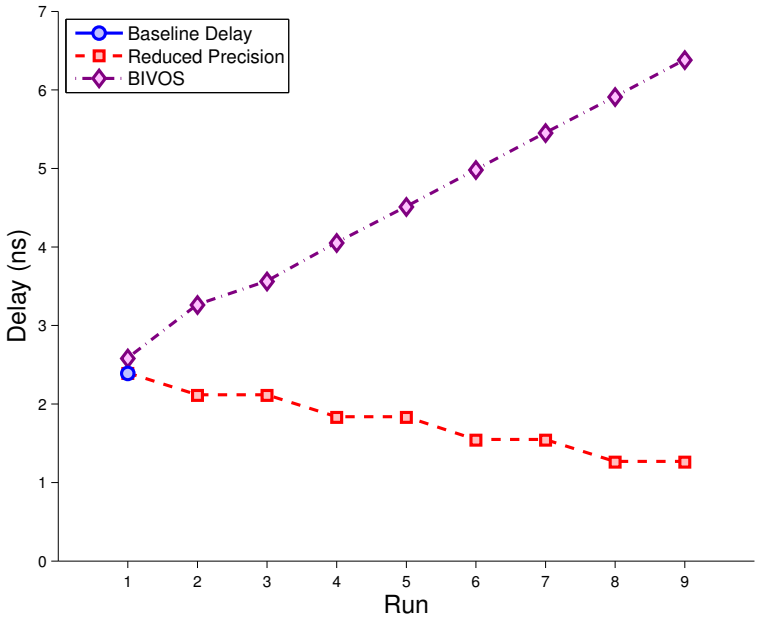


Figure 36: Worst case propagation delay for reduced-precision and inverter-based, BIVOS ripple-carry-adder implementations. Where the reduced-precision solution reduces delay by powering down circuit elements, the BIVOS solution increases delay through voltage scaling.

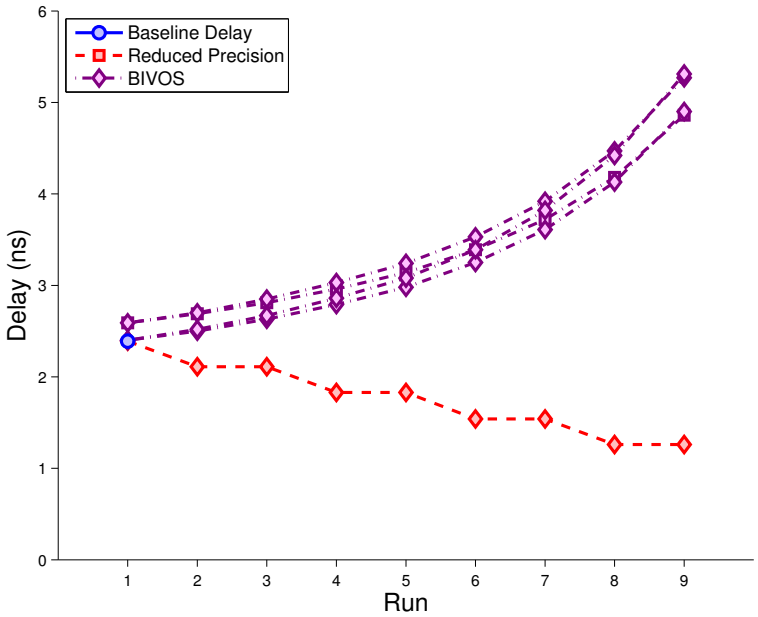


Figure 37: Worst case propagation delay for reduced-precision and level-converter-based, BIVOS ripple-carry-adder implementations. Again, the reduced-precision solution reduces delay by powering down circuit elements while the BIVOS solutions increases delay through voltage scaling.

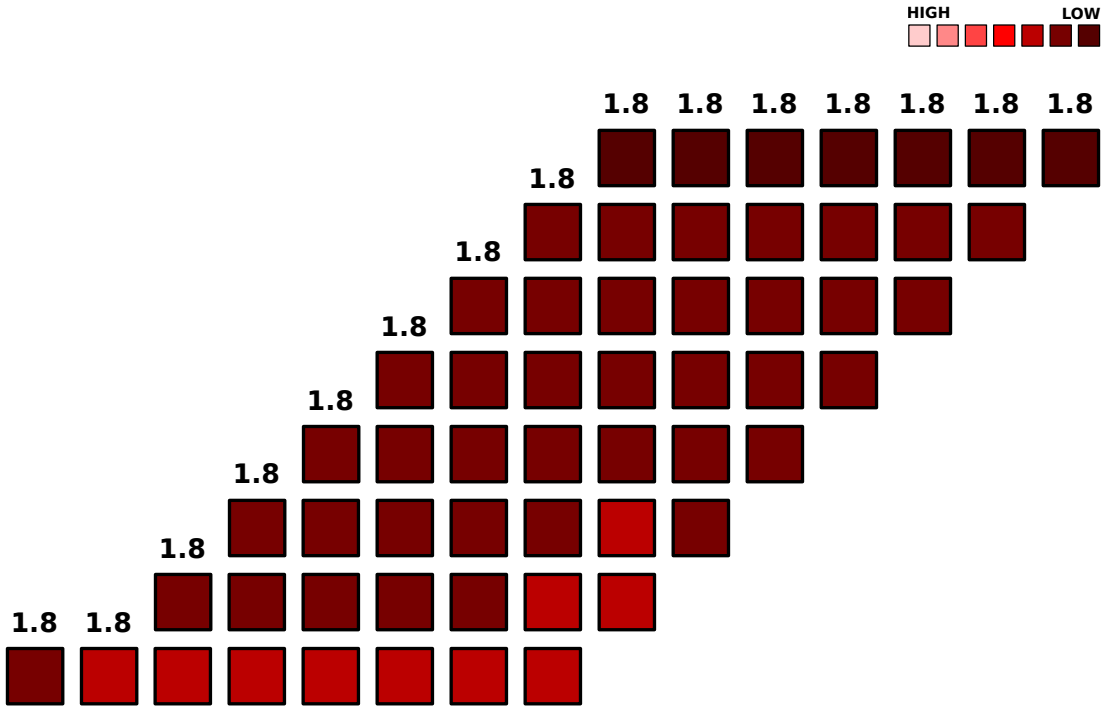


Figure 38: Switching activity for a standard CMOS, array multiplier operating at nominal voltage ($1.8V$). Unlike the ripple-carry adder, the design does not result in an alternating high-to-low activity pattern along positive and negative full-adder boundaries. This is due to input filtering applied by NAND gates that reduces the likelihood of input switching.

4.7.2 Array Multiplier

More complicated than the ripple-carry adder, the array multiplier consists of an eight-by-eight array of full adders forming a two dimensional structure. It was configured for 8-bit inputs resulting in 16-bit outputs. Shown in Figure 38, switching activity for a standard CMOS solution increases along the vertical dimension of the circuit with the highest switching activity occurring at the last circuit row. As was the case for the ripple-carry adder, these full adders are most likely to complete an initial switching on input transitions before switching again as delayed inputs arrive along propagation paths. Unlike the ripple-carry adder, however, the array multiplier does not display an alternating pattern of high-to-medium activity despite employing alternating logic. This is due to the NAND gates at each full-adder input filtering incoming circuit inputs, resulting in a reduced likelihood of input switching.

Figure 39 shows the switching activity for a reduced-precision solution with eight bits

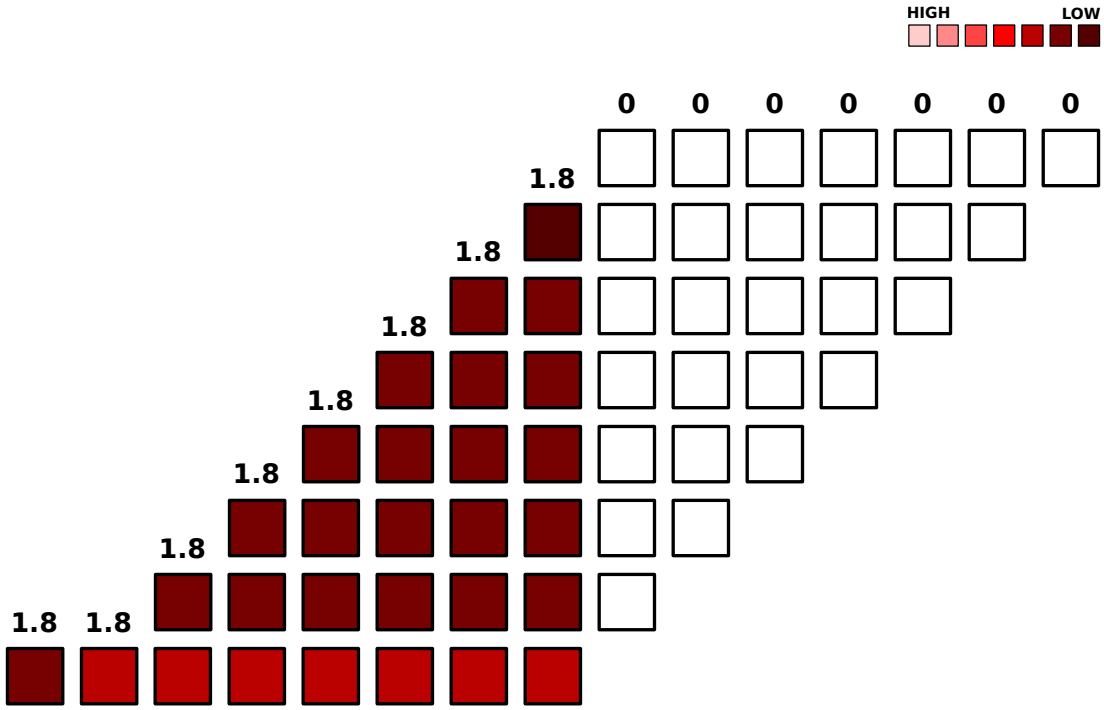


Figure 39: Switching activity for a standard CMOS, array multiplier using reduced-precision operation with eight bits powered down. The circuit exhibits switching activity that is similar to the nominal design at the bit positions that are powered.

powered down. Similar to the ripple-carry adder, activity for the reduced-precision implementation mimics the standard design with the exception of the powered-down full adders.

Figures 40 and 41 show the switching activity for the inverter-based and tradition level conversion BIVOS solutions respectively. Not unlike the ripple-carry adder solutions, the inverter-based design has the highest activity through the center of the circuit. Again, this is due to the propagation delay incurred from the bit positions biased near $0.8V$. The design employing traditional level conversion exhibits increased switching activity throughout much of the circuit when compared to a standard implementation. Increased switching activity at the high-order bit positions is once again expensive as this increases power consumption in full adders operating at nominal voltage.

Error distributions for the same extreme power-reduction designs employed previously are shown in Figure 42. As was the case for the ripple-carry adder, the reduced-precision solution concentrates a high frequency of errors at lower-order bit positions while the BIVOS solutions distribute far fewer errors over a larger number of bit positions. Unlike the adder,

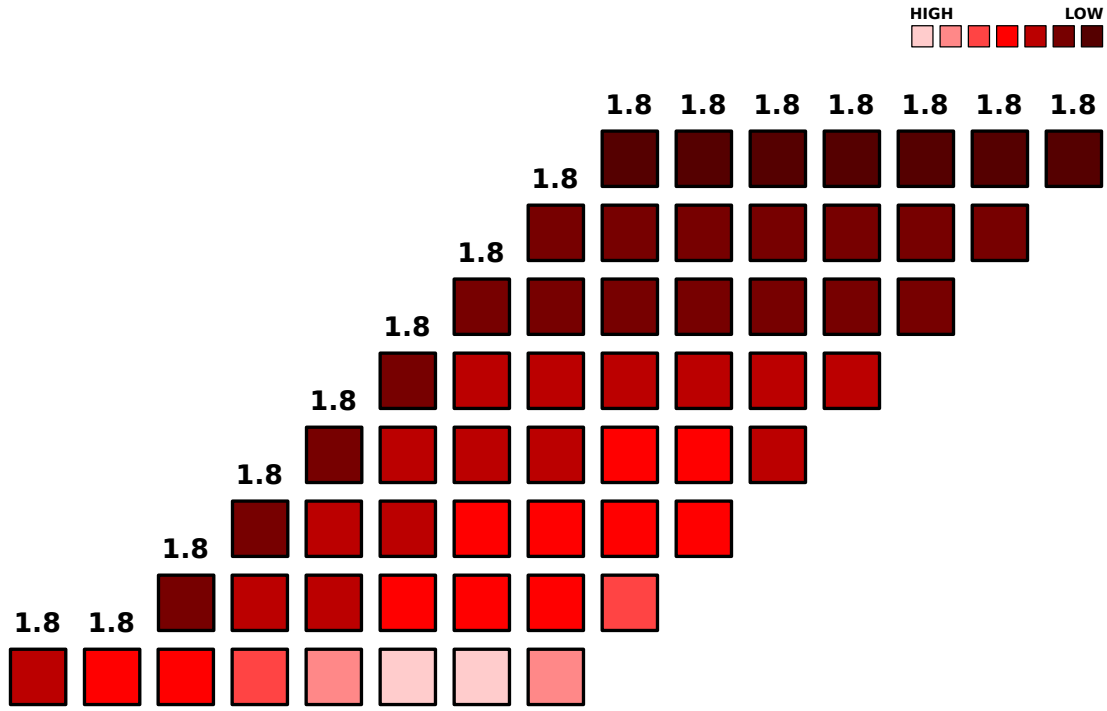


Figure 40: Switching activity for an array multiplier employing BIVOS with inverter-based level conversion with an initial bias position at bit 15. Delay incurred through the application of voltage scaling accumulates as additional switching activity along the last row of the circuit.

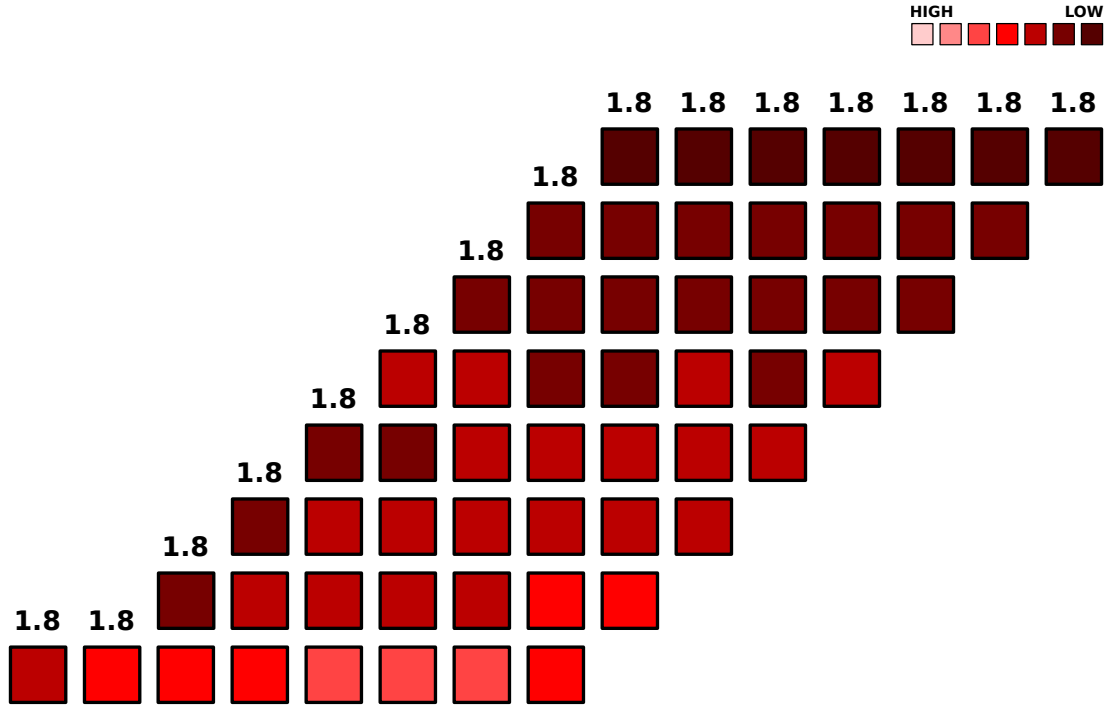


Figure 41: Switching activity for an array multiplier employing BIVOS with traditional level conversion with 13 bits biased to $1.0V$. Delay again accumulates as additional switching activity along the last row of the circuit.

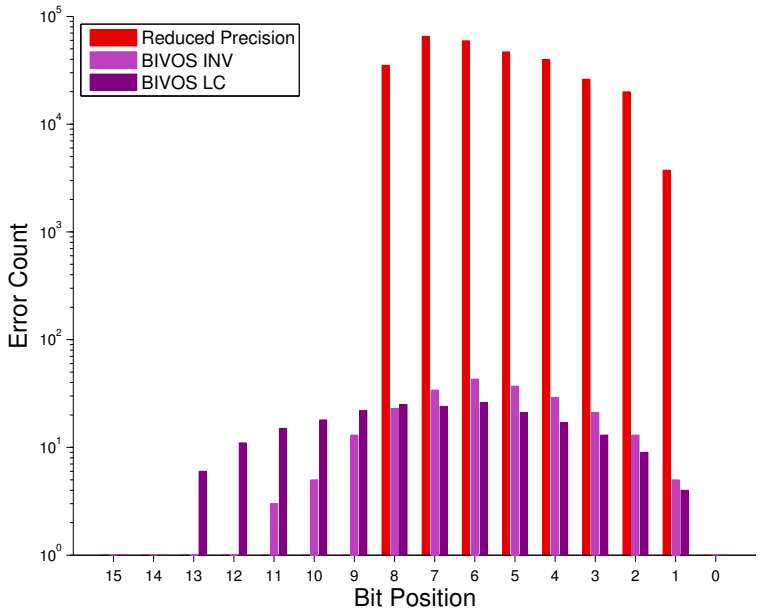


Figure 42: Bit-error rates by bit position for reduced-precision and BIVOS array-multiplier implementations. Again, the reduced-precision solution distributes a large number of errors over low-order bit positions while the BIVOS solutions distribute fewer errors over a larger range of bit positions. All solutions exhibit increased error counts through the center of the device as a result of the increased full-adder counts at these bit positions.

the multiplier solutions show a higher frequency of errors through the inner bit positions for all designs. This is due to the array-multiplier structure where inner bit positions have a higher number of full-adders allowing for more errors through the center of the structure.

Comparing MSE and switching energy for the reduced-precision and inverter-based BIVOS solutions in Figure 43, the BIVOS solution is unable to outperform the reduced-precision solution for any data points (in direct contrast to the inverter-based, BIVOS ripple-carry adder). Further, the inverter-based BIVOS solution is only capable of reducing power consumption below a standard implementation at three data points. This is due to the fact that the nominal solution is able to eliminate as many as three inverters at each full adder using alternating logic along the two dimensions of the multiplier structure. As a result, the BIVOS array multiplier is at a severe power disadvantage when utilizing inverters for level conversion given the design requires 122 more inverters than an alternating-logic implementation.

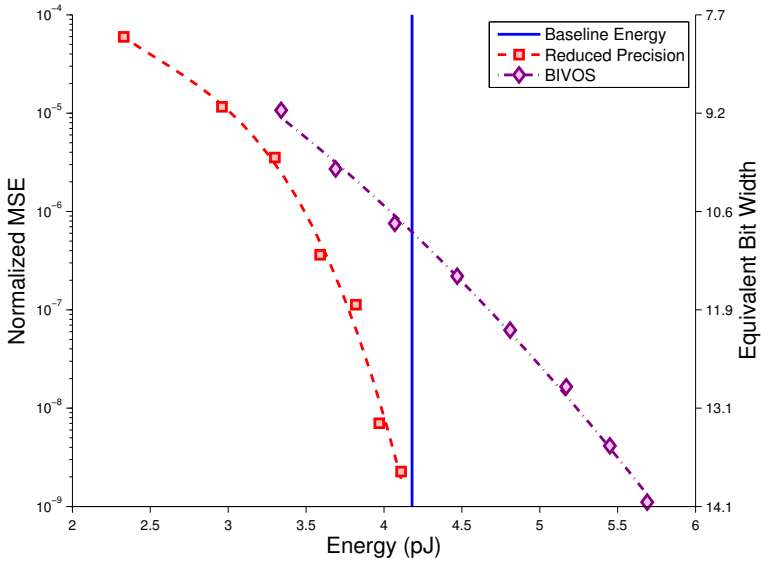


Figure 43: Mean-squared error vs energy for a fixed-point, array multiplier. Where the reduced-precision solution is able to reduce energy consumption compared to a standard CMOS array multiplier, a BIVOS design implemented with inverter-based level conversion suffers from a multitude of additional inverters and is unable to overcome the added power consumption. As a result, the BIVOS solution is only able to improve upon the energy consumption of a standard design at three data points.

BIVOS solutions employing traditional level conversion performed far better than the inverter-based alternative. Shown in Figure 44, BIVOS implementations reduced MSE when compared to a reduced-precision solution for most data points. More pronounced than the adder solutions, particularly with smaller biasing bin sizes, BIVOS configurations show the same diminishing return with increased voltage scaling. This is again due to increased switching activity (and the associated increase in energy consumption at unbiased bits) with greater and greater delay between propagation waves along voltage boundaries.

Unlike the ripple-carry adder, a reduced-precision multiplier solution yields little improvement in worst-case propagation delay (Figures 45 and 46). This is because the worst-case propagation path in the array multiplier is vertical along bit 7 and then horizontal across the final row. As a result, powering down full adders by column does not remove any from the critical path until bit 7 is powered-down. In the case of the BIVOS implementations, both the inverter and traditional level converter designs again suffer a performance penalty.

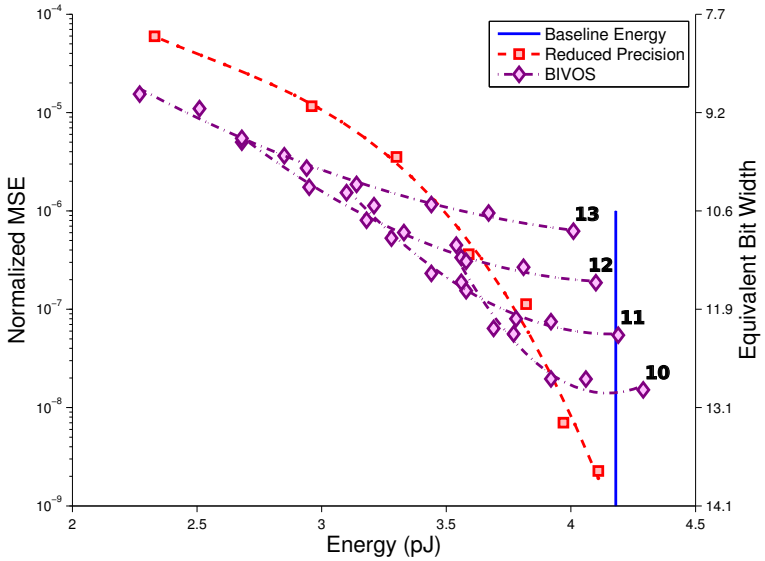


Figure 44: Mean-squared error vs energy for a fixed-point, array multiplier. Both a reduced-precision and BIVOS solutions employing traditional level conversion are able to improve on a standard CMOS design. Unlike the inverter-based designs, those utilizing traditional level conversion outperform a reduced-precision solution at multiple data points.

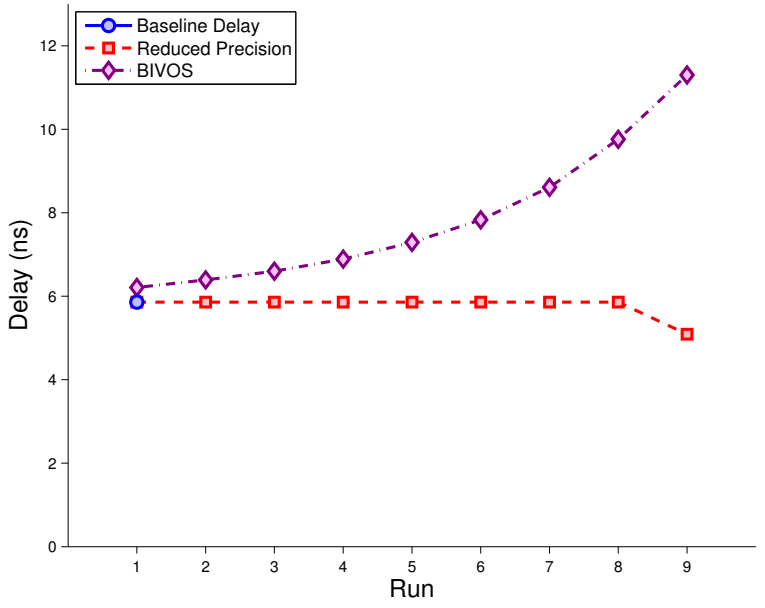


Figure 45: Worst case propagation delay for reduced-precision and inverter-based, BIVOS array-multiplier implementations. In this case, a reduced-precision design is unable to eliminate transistors for most data points due to a critical path that runs vertical at bit 7. As before, a BIVOS solution increases delay through voltage scaling.

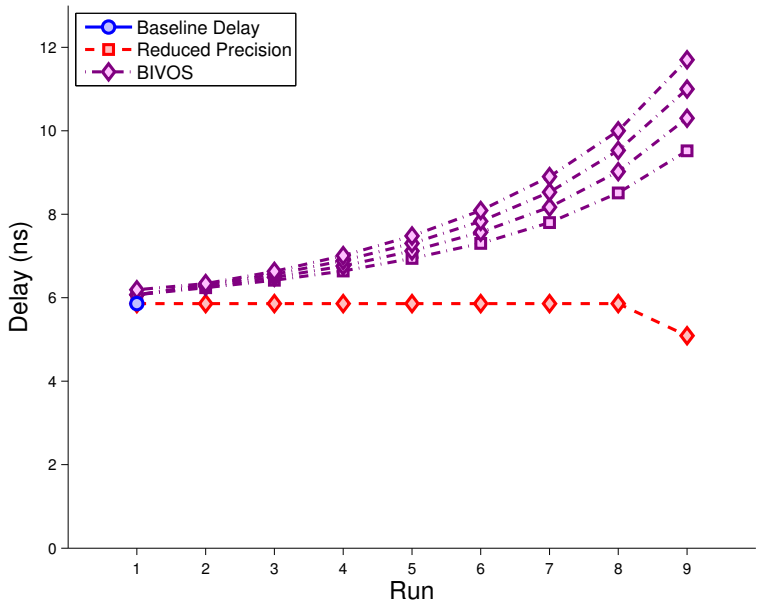


Figure 46: Worst case propagation delay for reduced-precision and level-converter-based, BIVOS array-multiplier implementations. Again, delay is relatively constant for a reduced-precision design where BIVOS increases delay due to voltage scaling.

4.7.3 Finite-Impulse-Response Filter

Finally, the most complicated circuit of the three was the FIR filter. It employed 6-taps using the same adder and multiplier structures tested previously. The circuit was simulated for three sets of coefficients (Table 10): low-pass filtering, high-pass filtering, and H.264 sub-pixel interpolation. Typical digital-signal processors will use two to four guard bits to "guard" against overflow in a series of mathematical operations (such as those present in an FIR filter). Utilizing guard bits, however, adds extra high-order bit positions, exacerbating the affect of propagation waves on power consumption. Instead, low and high-pass filter coefficients where chosen to ensure overflow would not occur (H.264 coefficients were defined as such) and the FIR filter was designed to utilize no guard bits. The resulting magnitude responses for the low and high-pass filter coefficients are shown in Figures 47 and 48 respectively.

Shown in Figure 49 for the low-pass filter, switching activity for the standard design is highest at the ripple-carry adders where the longest propagation paths terminate. This is the case for the reduced-power solutions as well, with the reduced precision solution roughly

Table 10: FIR Filter Coefficients

	Tap 0	**Tap 1**	**Tap 2**	**Tap 3**	**Tap 4**	**Tap 5**
Low-pass	−10	25	61	61	25	−10
High-pass	19	25	−64	63	−25	−19
Sub-pixel	1	−5	20	20	−5	1

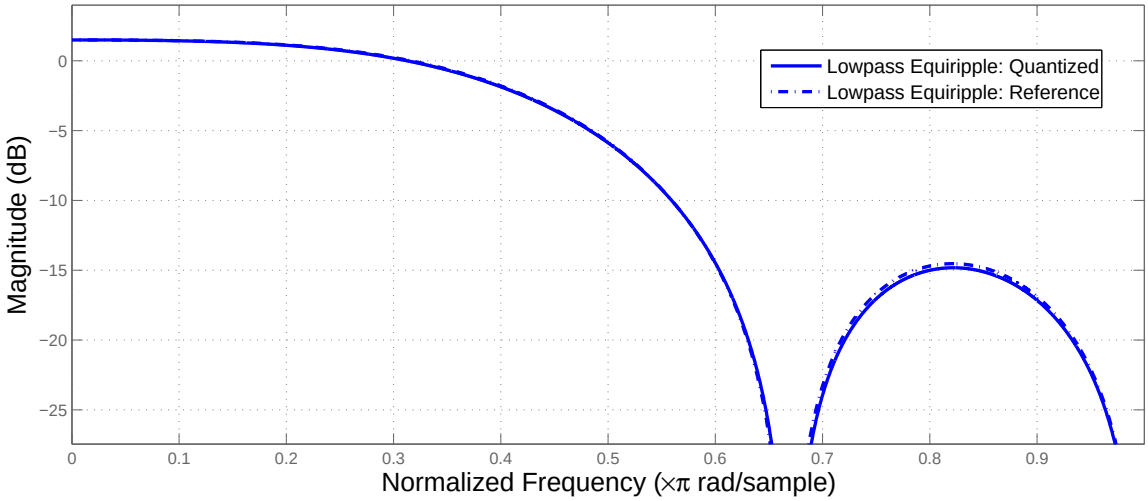


Figure 47: Magnitude response for an 8-bit, low-pass, FIR filter. Normalized frequencies below 0.7π are passed without attenuation while those above are suppressed.

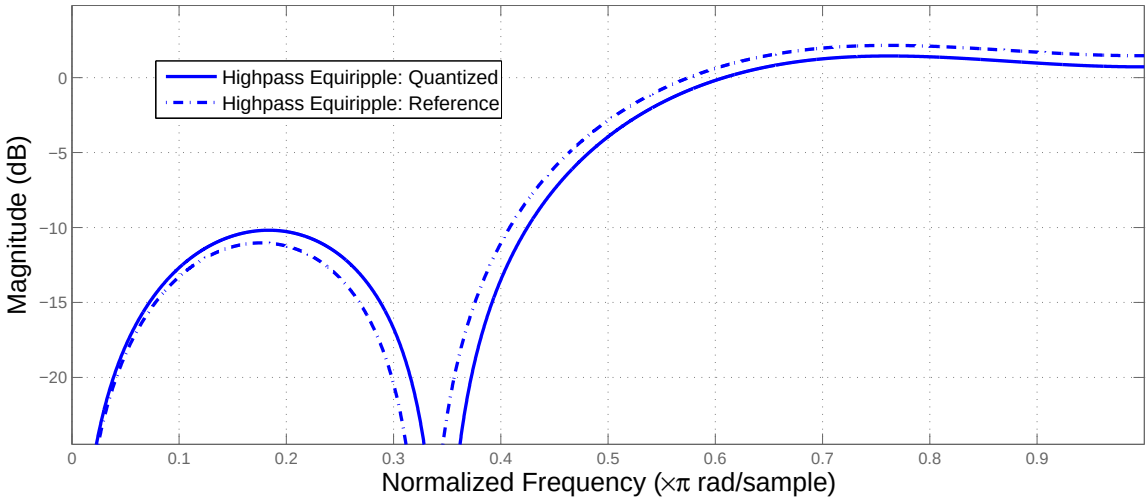


Figure 48: Magnitude response for an 8-bit, high-pass, FIR filter. Normalized frequencies above 0.3π are passed without attenuation while those below are suppressed.

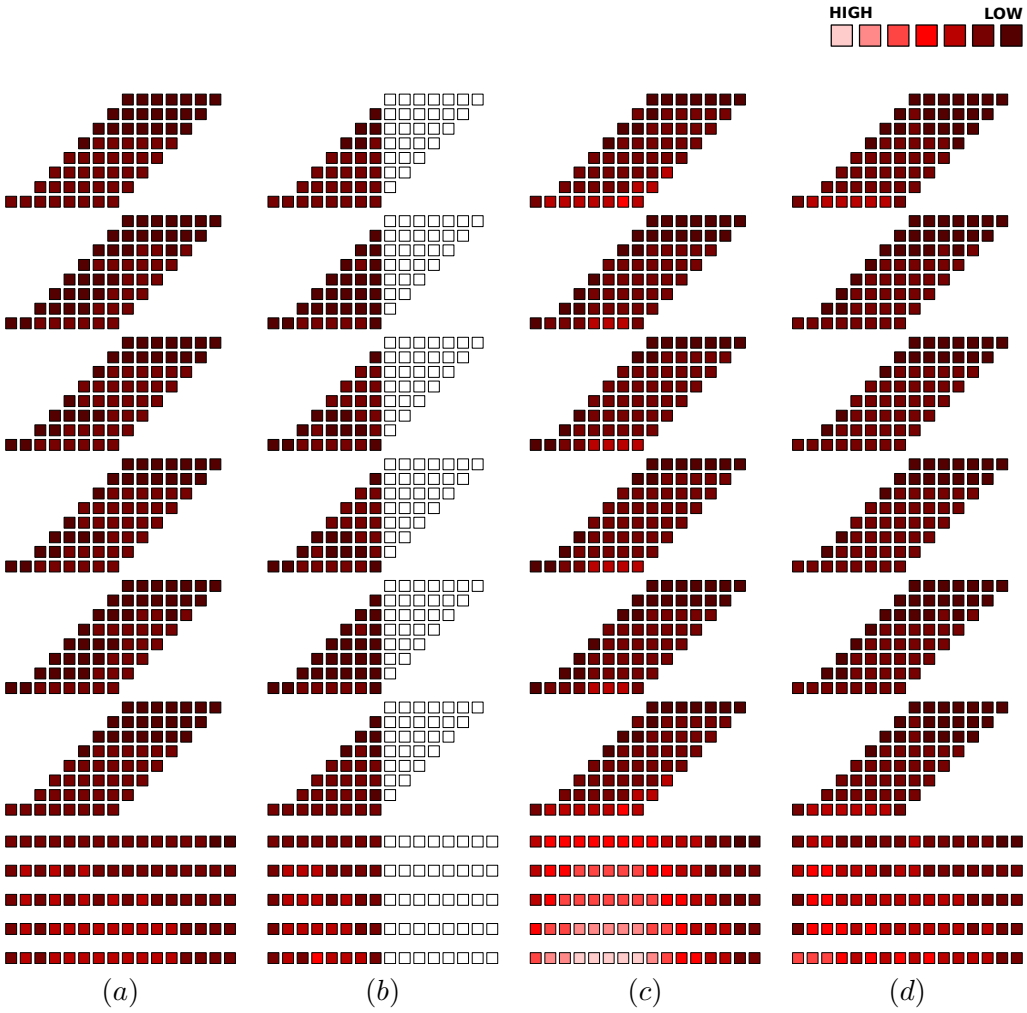


Figure 49: Switching activity for a low-pass FIR filter with four implementations: (a) standard CMOS $(1.8V)$, (b) reduced-precision with eight bits powered down, (c) BIVOS with inverter level conversion and an initial bias position at bit 15, and (d) BIVOS with traditional level conversion and 13 bits biased to $1.0V$. Delay incurred through voltage biasing results in increased switching activity at the ripple-carry-adder outputs.

mirroring standard CMOS activity. As was the case for the adder and multiplier circuits, switching activity is increased for the BIVOS designs. Again, the inverter-based design shows the highest activity at inner bit positions and the design employing tradition level conversion suffers at high-order bit positions.

Error distributions are shown in Figures 52, 53, and 54 for the low-pass, high-pass, and sub-pixel-interpolation filters respectively. While there is variation due to filter coefficients, the distributions are roughly equivalent. As the array multipliers comprise the majority of the filter circuitry, the error distributions show the same higher frequency of bit errors at

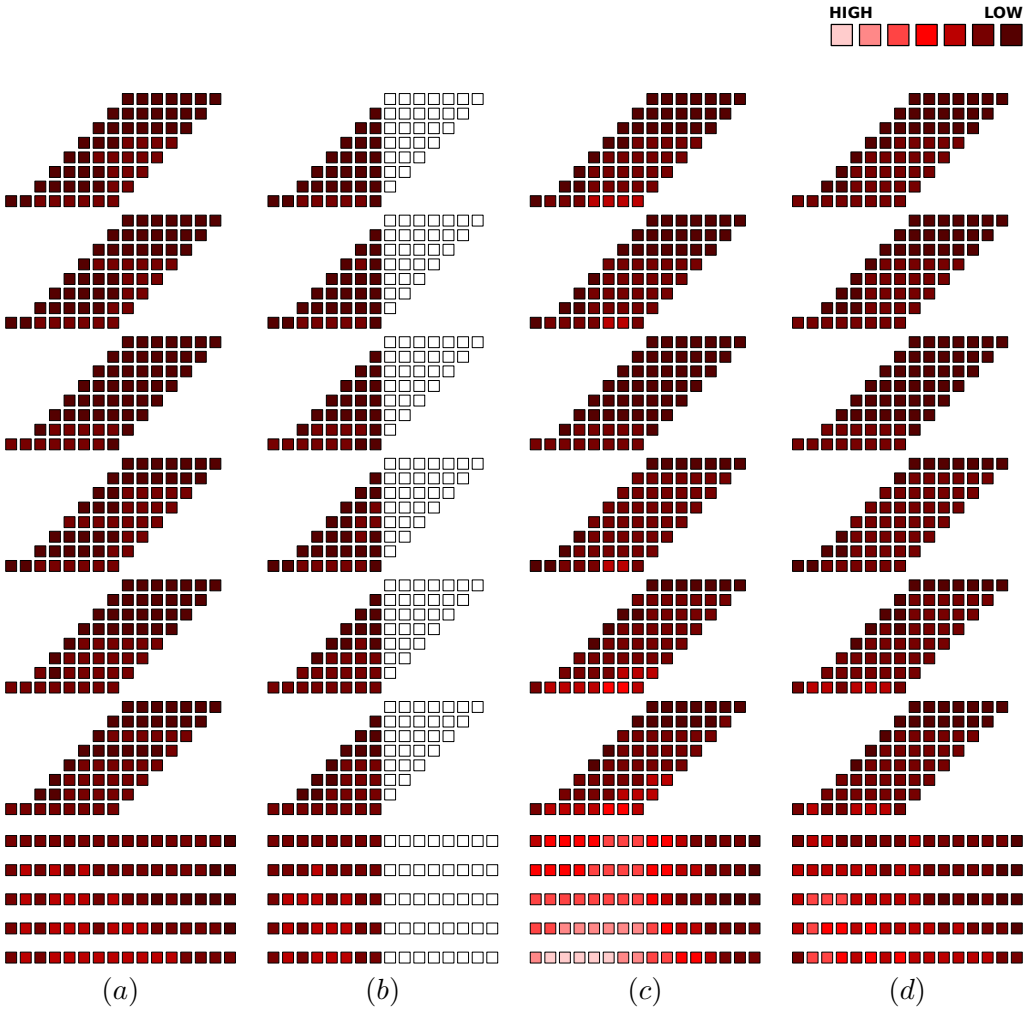


Figure 50: Switching activity for a high-pass FIR filter with four implementations: (a) standard CMOS ($1.8V$), (b) reduced-precision with eight bits powered down, (c) BIVOS with inverter level conversion and an initial bias position at bit 15, and (d) BIVOS with traditional level conversion and 13 bits biased to $1.0V$. Again, delay incurred through voltage biasing results in increased switching activity at the ripple-carry-adder outputs.

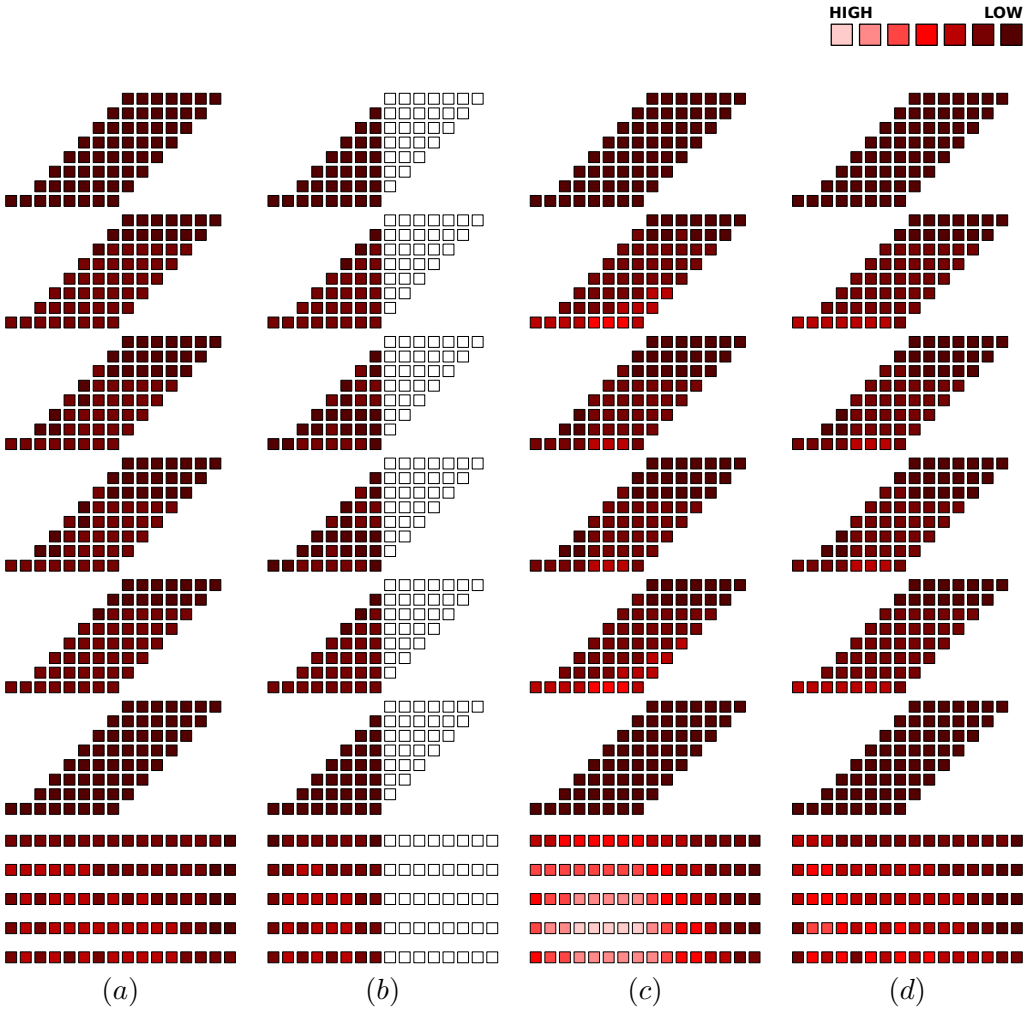


Figure 51: Switching activity for a sub-pixel-interpolation, FIR filter with four implementations: (a) standard CMOS ($1.8V$), (b) reduced-precision with eight bits powered down, (c) BIVOS with inverter level conversion and an initial bias position at bit 15, and (d) BIVOS with traditional level conversion and 13 bits biased to $1.0V$. As before, delay incurred through voltage biasing results in increased switching activity at the ripple-carry-adder outputs.

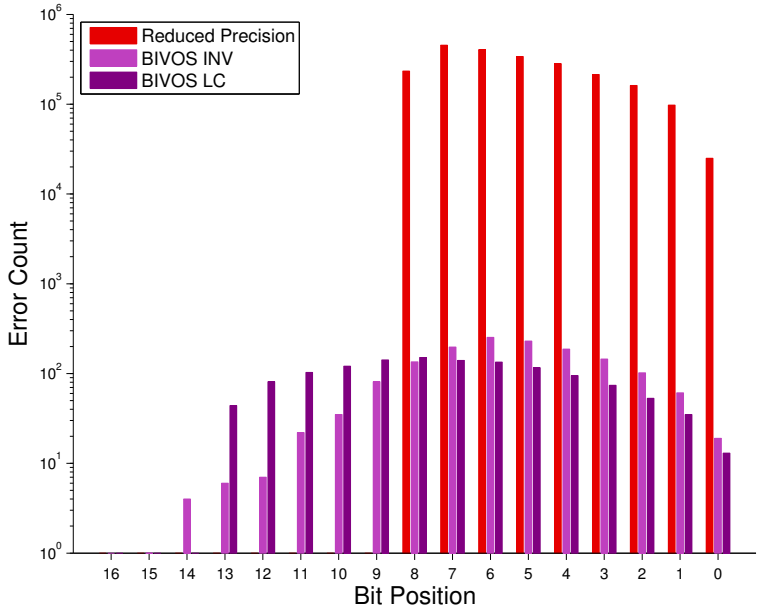


Figure 52: Bit-error rates by bit position for reduced-precision and BIVOS, low-pass FIR-filter implementations. As the circuit structure of the FIR filter is dominated by array multipliers, the error rates follow a pattern similar to that found in the array multiplier.

inner bit positions as were present in the multiplier designs. Similar to the other circuits, reduced-precision solutions yield a large quantity of bit errors, with limited magnitude, while BIVOS solutions distribute fewer errors across more bit positions.

Again, as array multipliers dominate the filter circuitry, MSE and switching energy are largely determined by multiplier configurations. This is despite the fact that the highest switching activity within the filter circuit is at the ripple-carry adders. Evident from Figures 55, 56, and 57; the inverter-based level conversion that yields highly favorable results for a ripple-carry adder is unable match reduced-precision solutions for the FIR filter. Instead, the resulting MSE versus switching energy plots more closely resemble those of the array multiplier.

Shown in Figures 58, 59, and 60; the BIVOS solutions employing traditional level conversion also closely resemble those of the array multiplier. As was the case for the multiplier, the traditional-level-conversion solutions outperform reduced-precision solutions at many data points. The trade-off between accuracy and delay is strongly pronounced for the FIR filter solutions. At the smallest bin sizes, the increase in high-order power consumption

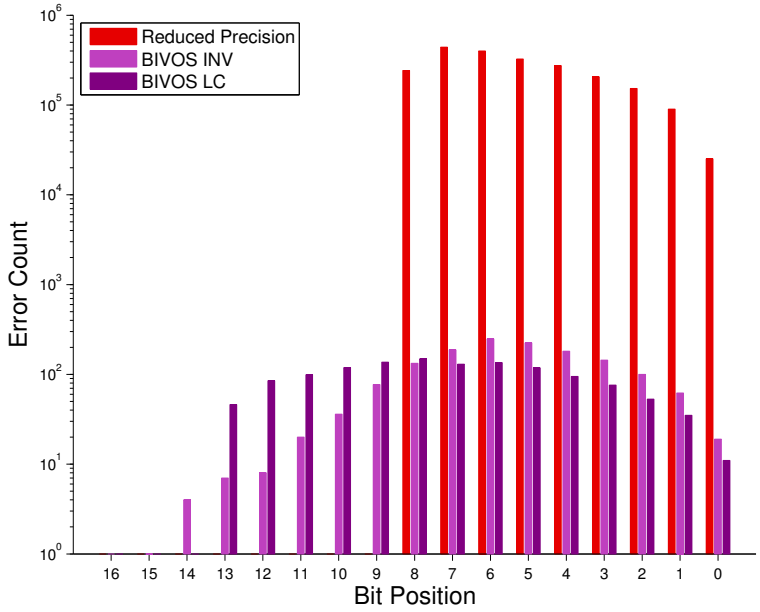


Figure 53: Bit-error rates by bit position for reduced-precision and BIVOS, high-pass FIR-filter implementations. Again, the error rates follow a pattern similar to that found in the array multiplier.

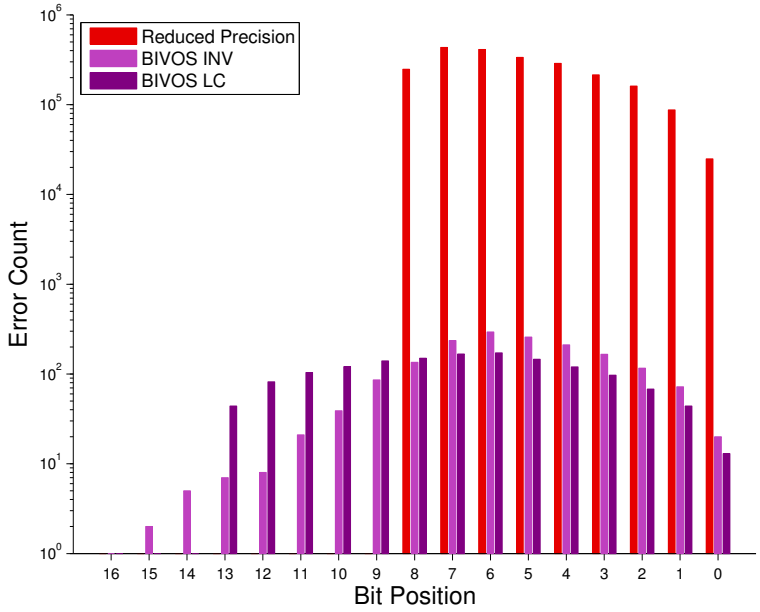


Figure 54: Bit-error rates by bit position for reduced-precision and BIVOS, sub-pixel-interpolation FIR-filter implementations. Once again, the error rates follow a pattern similar to that found in the array multiplier.

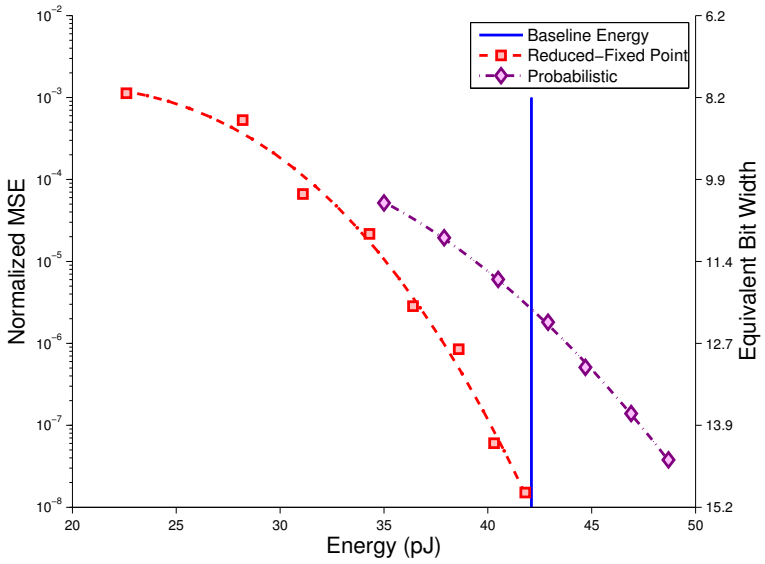


Figure 55: Mean-squared error vs energy for reduced-precision and BIVOS (implemented with inverter level conversion) low-pass, FIR filter implementations. Similar to the array multiplier, an inverter-based solution is unable to outperform a reduced-precision solution.

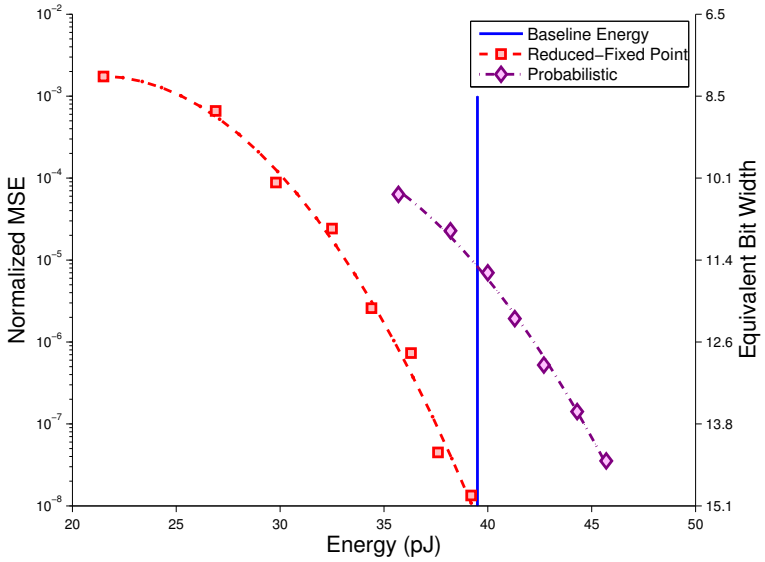


Figure 56: Mean-squared error vs energy for reduced-precision and BIVOS (implemented with inverter level conversion) high-pass, FIR filter implementations. Again, an inverter-based solution is unable to outperform a reduced-precision solution.

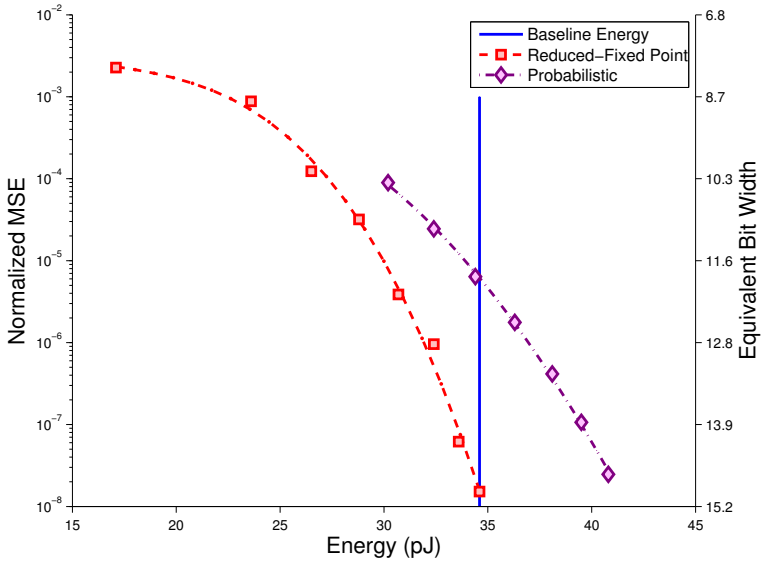


Figure 57: Mean-squared error vs energy for reduced-precision and BIVOS (implemented with inverter level conversion) sub-pixel-interpolation, FIR filter implementations. Once more, an inverter-based solution is unable to outperform a reduced-precision solution.

is so significant that the lowest-voltage (and highest MSE) solutions require more energy than others operating at a higher voltage. Despite this trade-off, the overall trend indicates that traditional level conversion is superior to other designs across the three sets of filter coefficients.

Worst case propagation delay for the FIR filter is shown in Figures 61 and 62. Since the constant filter coefficients utilized for low-pass, high-pass, and H.264 filtering impact the ability to activate the critical path, worst case propagation delay was estimated as the summation of the worst case multiplier delay and adder delay at each configuration. Delay was estimated for the FIR filter in general, as filter coefficients were ignored in the calculation. Due to the inclusion of ripple-carry adders, reduced-precision solutions once again decreased overall propagation delay. BIVOS solutions, in turn, increased worst-case propagation delay with increased voltage scaling.

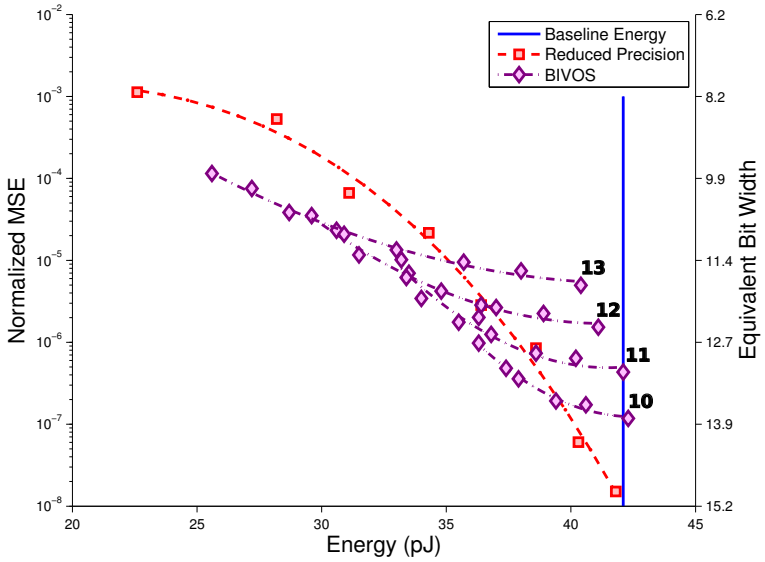


Figure 58: Mean-squared error vs energy for reduced-precision and BIVOS (implemented with traditional level conversion) low-pass, FIR filter implementations. As was the case for the multiplier, BIVOS designs employing traditional level conversion outperform a reduced-precision design at a majority of the data points tested.

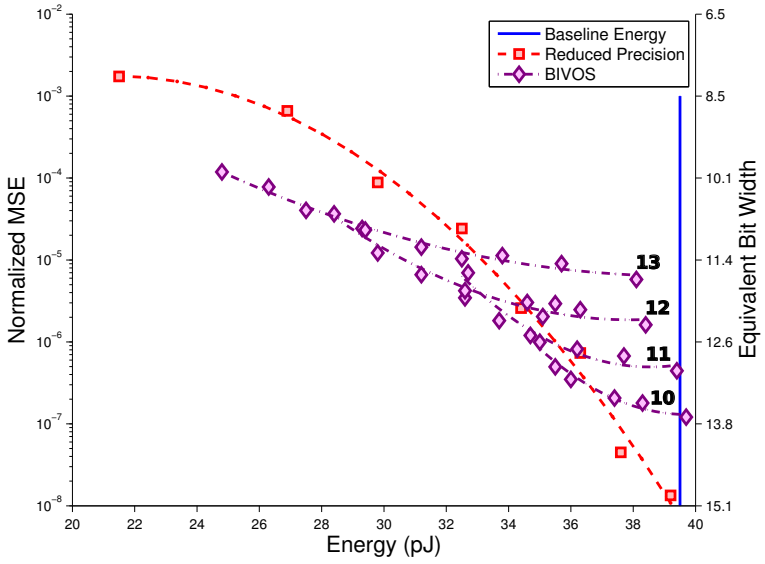


Figure 59: Mean-squared error vs energy for reduced-precision and BIVOS (implemented with traditional level conversion) high-pass, FIR filter implementations. Again, BIVOS designs employing traditional level conversion outperform a reduced-precision design at a majority of the data points tested.

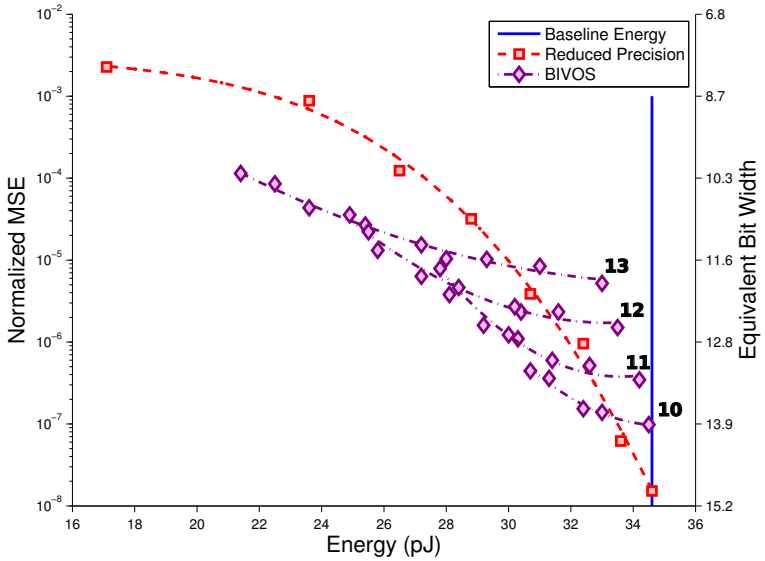


Figure 60: Mean-squared error vs energy for reduced-precision and BIVOS (implemented with traditional level conversion) sub-pixel-interpolation, FIR filter implementations. As before, BIVOS designs employing traditional level conversion outperform a reduced-precision design at a majority of the data points tested.

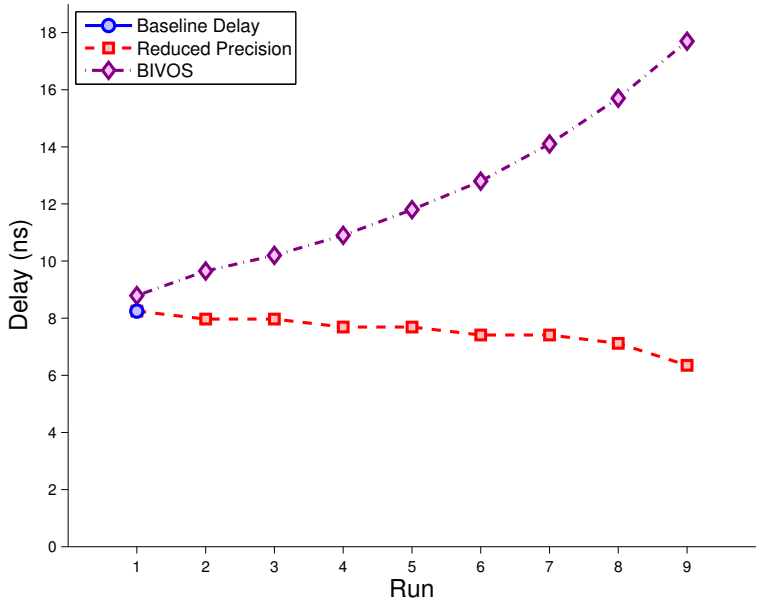


Figure 61: Worst case propagation delay for reduced-precision and inverter-based, BIVOS FIR-filter implementations. Operating as a combination of adder and multiplier elements, the reduced-precision design decreases propagation delay by powering down circuit elements. The BIVOS solution, conversely, increases delay through voltage scaling.

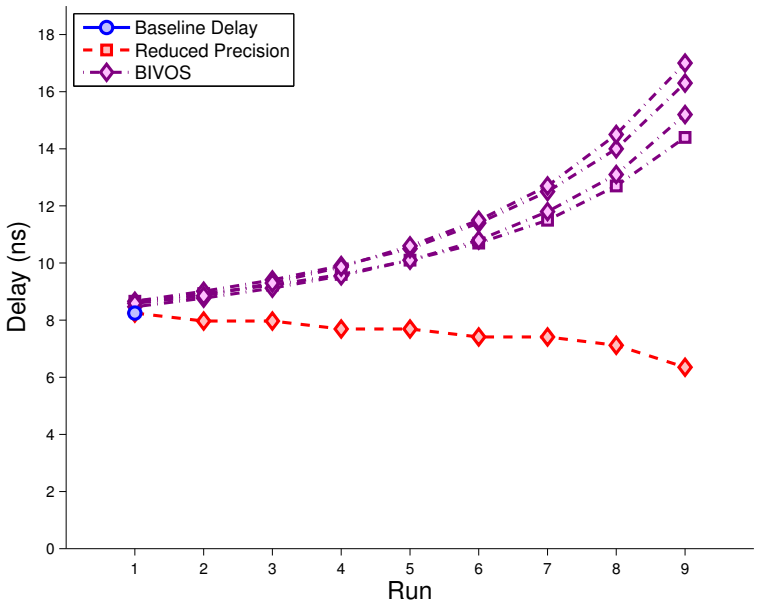


Figure 62: Worst case propagation delay for reduced-precision and level-converter-based, BIVOS FIR-filter implementations. Again, the reduced-precision design decreases propagation delay by powering down circuit elements and the BIVOS solution increases delay with voltage scaling.

CHAPTER V

H.264 VIDEO DECODING AS A PROOF OF CONCEPT

5.1 H.264 Video Decoding

Video decoding, specifically H.264 video decoding, was chosen as an application to evaluate the effectiveness of BIVOS as a technique. Discussed earlier, video decoding falls into the category of resilient applications since it is ultimately a human viewer who determines application quality. H.264 video decoding was selected based on the common use of the specification in platforms ranging from broadcast television to Internet and cellular streaming.

H.264 is a hybrid video coding standard that compresses data by reducing redundancies: spatial, temporal, perceptual, and statistical [34]. Video is encoded as a sequence of pictures, or frames. Each frame is subdivided into small 16x16 blocks of pixels and each pixel is encoded using the YCbCr (luma Y, or brightness, and chroma Cb/Cr, or color deviation from gray toward blue/red) color space. While the pixel data in some blocks is encoded directly, compression is achieved by predicting the majority of the pixel data based previously encoded blocks [87]. The resulting block prediction data is then compressed using an integer transform and quantized. Finally, blocks are grouped into frames to complete the encoding.

Decoding is comprised of a five step process, shown in Figure 63, that reverses the encoding process. The first step in the decoding process is obtaining the information necessary to reconstruct the frame (block type, quantizer parameters, reference frame indexes, etc.) in the entropy decoder [71]. From the entropy decoder, the inverse quantizer reverses the quantization process employed in the encoder and the inverse transform reverses the compression transform used in encoding. Once the original block data is decoded, it is passed to the motion compensation stage where pixel data is retrieved using prediction data based on previously decoded blocks. If the prediction data is based on blocks in the current frame

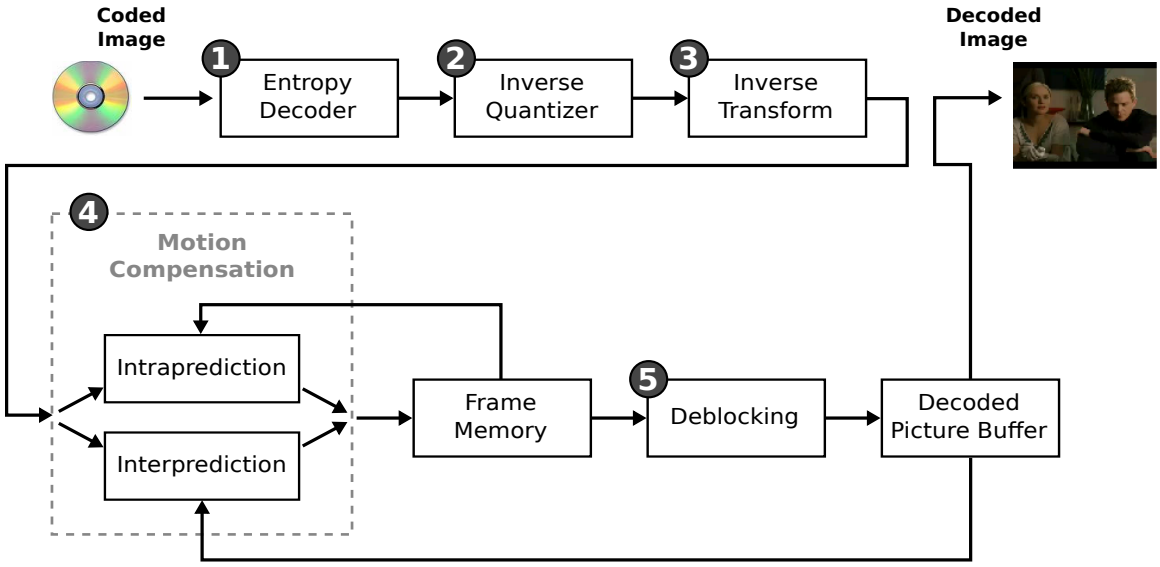


Figure 63: Flow chart of the five-stage H.264 decoding algorithm. 1. Entropy Decoder: decodes frame parameters. 2. Inverse Quantizer: reverses encoding quantization of block prediction data. 3. Inverse Transform: reverses compression transform applied to original block prediction data. 4. Motion Compensation: calculates block-pixel values by performing motion based prediction. Results are stored in frame memory for further intraprediction until the current frame is fully decoded. 5. Deblocking: Decoded blocks are filtered to smooth transitions across block edges. Resulting frame is stored in the decoded picture buffer for display and use in interprediction.

only, pixel data is retrieved using the intraprediction stage. If prediction data is based on blocks in previously decoded frames, pixel data is retrieved using the interprediction stage. Once all blocks have been decoded and the entire frame has been recovered, a filter is applied to reduce distortion along block edges in the deblocking stage [34, 71].

Within the motion compensation stages, motion resolution is defined to quarter-pixel resolution for luma samples. To achieve sub-pixel resolution, interpolation is used. First, half-pixel values are generated by interpolation of neighboring integer-pixels using a six-tap, finite-impulse-response (FIR) filter, shown in Figure 64. Linear interpolation is then used to generate quarter-pixel values by comparing neighboring half and integer-pixels [71, 98]. Operating on a RISC processor, such as an ARM processor, the sub-pixel computations typically account for more than 50% of computational time [98].

Within the H.264 algorithm, motion compensation was chosen for BIVOS operation (specifically the six-tap FIR filter was implemented using BIVOS). Because of the high

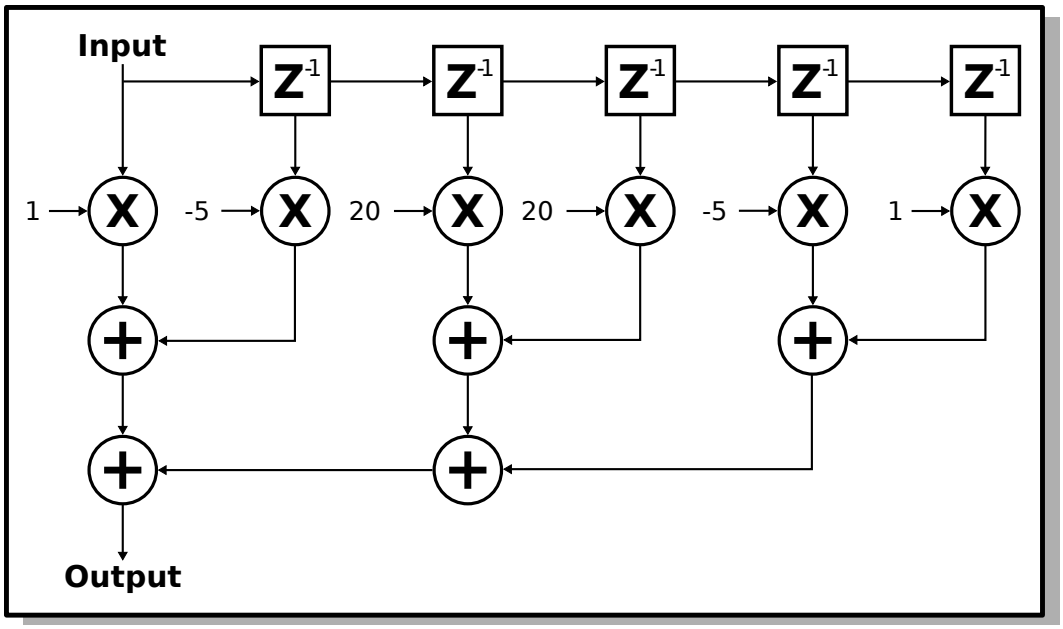


Figure 64: Six-tap, finite impulse response (FIR) filter used for sub-pixel interpolation in H.264 video decoding. The filter is comprised of delay (Z^{-1}), multiplication (X), and addition (+) elements. Interpolation is performed using the coefficients 1, -5, 20, 20, -5 and 1.

utilization of the motion compensation stage, it is an excellent candidate for power savings within the algorithm. In addition to comprising a computationally significant portion of video decoding, motion compensation is followed by deblocking that smooths irregularities along block borders. In the context of PCMOS, this deblocking process smooths small errors introduced through BIVOS and provides error masking that improves perceptual quality.

5.2 FIR Architecture

To evaluate the area impact of a BIVOS implementation layout was performed for both a BIVOS and a standard CMOS FIR filter design. In each case the motion-compensation FIR accepts color intensity (an eight-bit input) and employs both positive and negative coefficients for filtering. As such, the design calls for a 9-bit (8 bits for color intensity and an extra bit to maintain sign), fixed-point FIR filter.

Similar to Section 4.1, design was first performed for a standard CMOS implementation. The multiplier architecture utilizes a 9-bit-in (18-bit-out) array multiplier (Figure 65). As the FIR filter design is intended solely for H.264 video decoding, the B inputs on each of

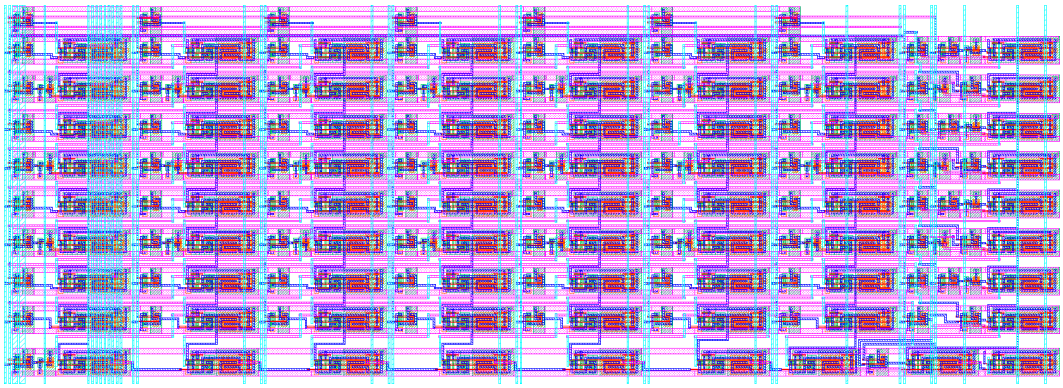


Figure 65: Standard CMOS implementation for an 8-bit-in, 16-bit-out, array multiplier. Circuit inputs and outputs are routed to vertical edges in metal three.

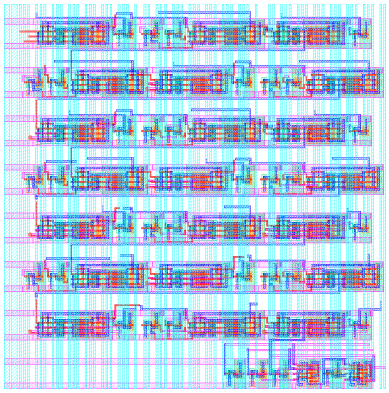
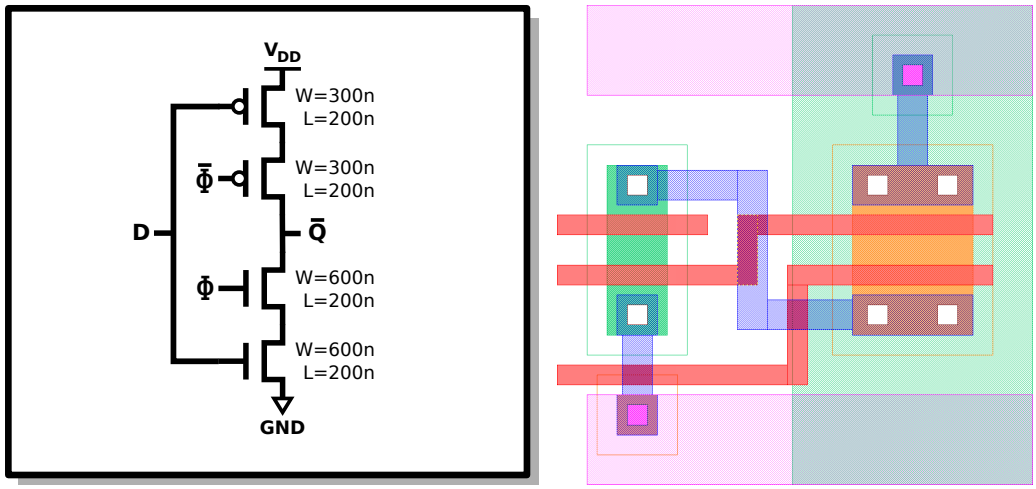


Figure 66: Standard CMOS implementation for an 18-bit ripple-carry adder. Circuit inputs and outputs are routed to vertical edges in metal three.

the six multipliers are hardwired to the appropriate coefficient values using V_{dd} and Gnd.

The adder architecture was implemented as an 18-bit-in (19-bit-out) ripple-carry adder (Figure 66). Designing for 18-bits allowed no guard bits to prevent overflow, however, the H.264 decoding coefficients ensure overflow will not occur. As overflow is guaranteed not to occur, bit 19 was discarded from each full adder.

The delay elements were implemented as a series of d-type flip-flops and each device required a pair of latches. Both the flip flops and the latches were designed using the Weste text as a basis [86]. Latch transistors were sized at three times a standard inverter for added drive strength with balanced rise and fall times (Figure 67). The flip-flop was then comprised of a pair of latches along with inverters to form the memory element (Figure 68). As $\overline{Q}$ was redundant, only Q was provided for the design. The flip-flop was sized to fit



(a) (b)

Figure 67: Latch implementation: (a) transistor schematic with transistor sizing for improved drive strength and (b) VLSI layout with cell pitch set 4.8μm resulting in a width of 4.45μm and an area of 21.36μm^2.

within the width of a full adder so that delay elements could be placed directly above the multiplier elements they were intended to drive.

The taps were arranged using a three row, two column layout with multipliers forming the outer edges and adders placed between (Figure 69). Delay elements were inserted between each multiplier and the space left from an uneven number of adders was utilized for the output buffer. A mesh network was utilized for clock distribution to minimize line resistance and clock skew [25]. With each clock pulse, input data moves from the input buffer above tap zero, down through tap two, across to tap three, and back up to tap five before exiting the filter. Between pulses, coefficients are applied to filter inputs through the multipliers at each tap position. Results are combined through the adders in the middle of the circuit and filter results are buffered for output at the bottom of the circuit. The resulting FIR filter is shown in Figure 70

FIR filter design was then repeated for a BIVOS implementation. The BIVOS multiplier was again designed as a 9-bit-in (18-bit-out) array multiplier and, based on results from Section 4.7, utilizes traditional level conversion with biasing split into two voltage bins (Figure 71). The low-order, biasing, bin was sized at 13 bits and the remaining 5 bits were placed in the high-order bin. As was the case for the standard CMOS design, filter

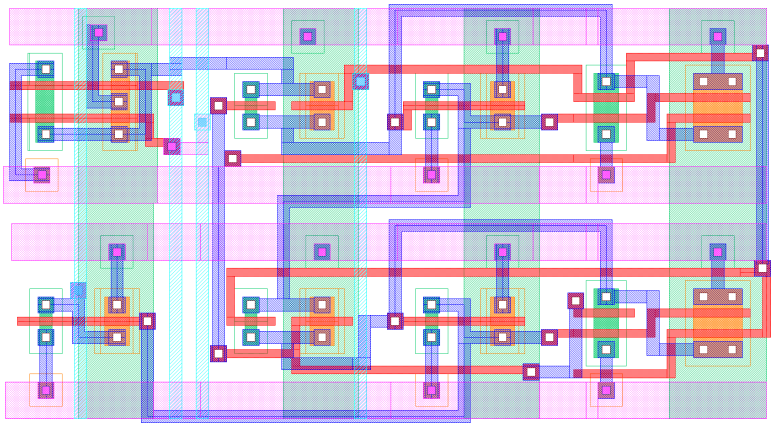


Figure 68: Standard CMOS implementation of a D-type flip-flop employing two latches along with several inverters to implement a memory element. The cell footprint has been sized to roughly fit within the width of a standard-cell full adder to allow for vertical alignment during layout.

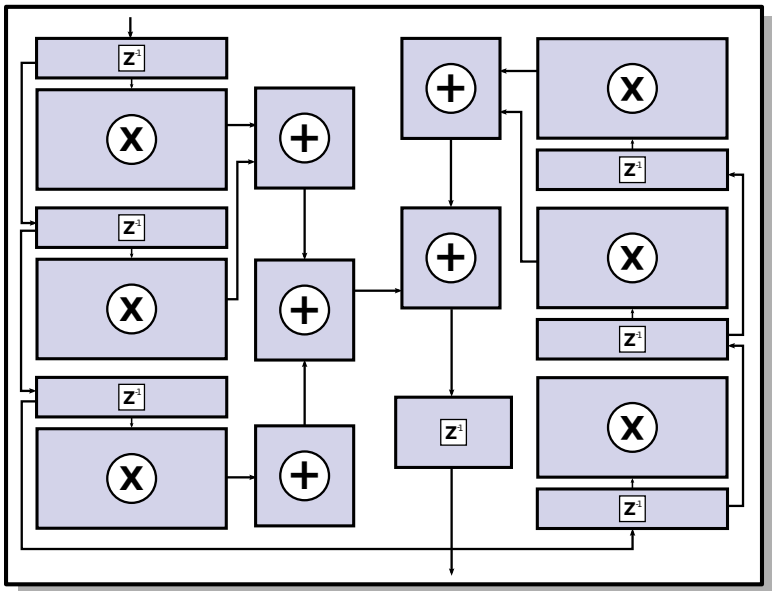


Figure 69: Floor plan, including signal routing, for the FIR filter layout. Circuit inputs enter at the upper left and propagate through the six tap position at each clock cycle. Tap inputs are multiplied by filter coefficient along the outer edges of the circuit and accumulate along the center of the circuit. Filtering results are then buffered before outputs exit through the lower edge of the circuit.

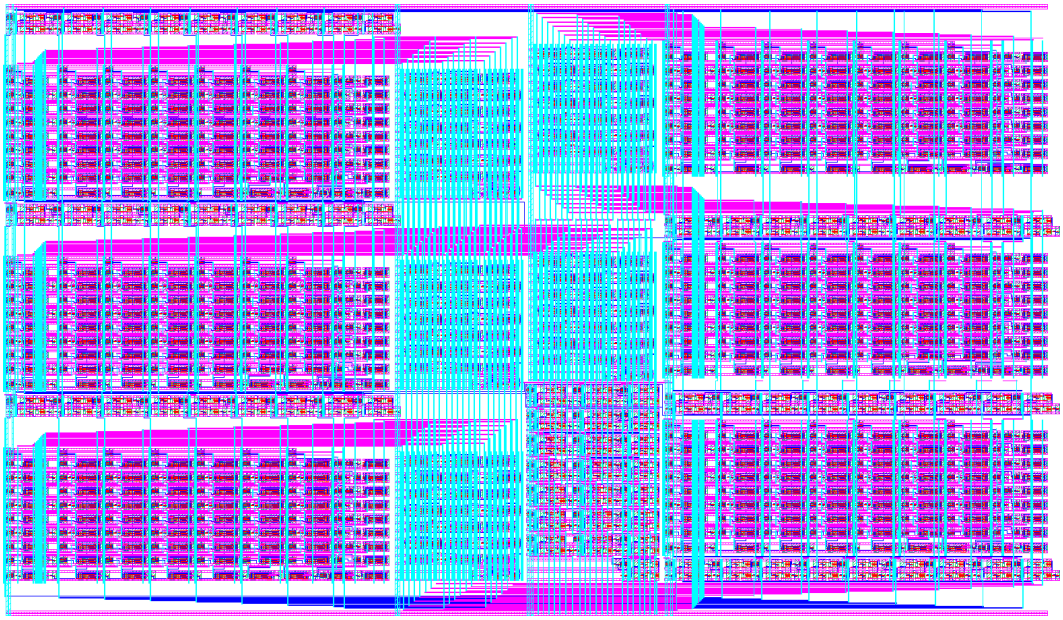


Figure 70: Standard CMOS implementation of a 9-bit, FIR filter as required for H.264 video decoding. The resulting design is roughly square at $506.3\mu m$ by $292.3\mu m$ and a total area of $0.147mm^2$.

coefficients are hardwired to B inputs on circuit multipliers.

Similar to the standard CMOS design, addition was implemented using a 18-bit-in (19-bit-out) ripple-carry adder with no guard bits. As was the case for the multiplier, standard level converters were used for voltage conversion. The design was again divided into two voltage bins along bit 13. As was the case for the standard CMOS design, sign extension was used for inputs that did not match data width. The resulting adder is shown in Figure 72.

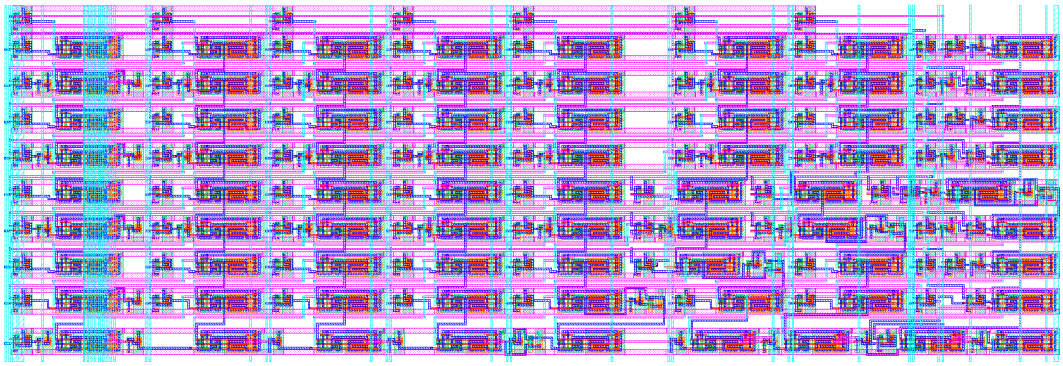


Figure 71: BIVOS implementation for an 9-bit-in, 18-bit-out, array multiplier. Circuit inputs and outputs are routed to vertical edges in metal three.

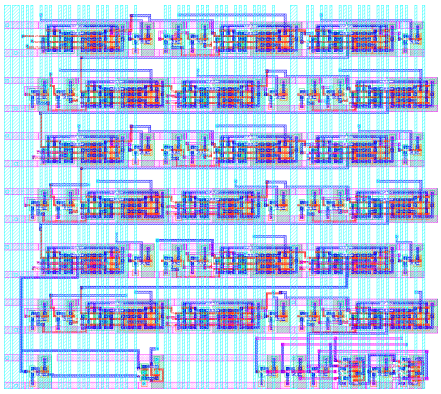


Figure 72: BIVOS implementation for an 18-bit ripple-carry adder. Circuit inputs and outputs are routed to vertical edges in metal three.

Layout for the BIVOS FIR filter employed the same floor plan used for the standard CMOS design with the added requirement of two voltage planes necessary for biasing. As each cell (adders and multipliers) routed biasing voltages vertically in metal three, cells only required horizontal alignment for vertical voltage distribution. In addition to the vertical distribution provided by the adder and multiplier cells, the two voltage lines were routed horizontally along the top and bottom of the circuit to join the voltage planes in each of the four columns. Layout for the resulting BIVOS FIR filter design is shown in Figure 73.

5.3 Generation of Multiple Voltage Levels

By definition, BIVOS requires multiple voltage sources for operation. As many modern embedded systems also require multiple voltage levels (although not to the extent required by BIVOS), there are existing techniques for providing the necessary voltages. Typically system energy comes from an off-chip power source. As power supply lines can be noisy and the supplied voltage rarely matches the requirements for system operation, voltage conversion is necessary. DC-DC converters transform this noisy input voltage into the required output voltage, monitoring varying system loading in the process and regulating the output voltage as needed.

Outlined in [48], there are several techniques for providing voltage regulation. Linear regulators monitor output voltage for deviations and adjust supply current as necessary to maintain proper output voltage under varying supply loads. By design they can only

Figure 73: BIVOS implementation of an 9-bit FIR filter as required for H.264 video decoding. The resulting design is roughly square at $514.4\mu m$ by $294.7\mu m$ and a total area of $0.151mm^2$.

provide step-down voltage conversion, are incapable of dynamic voltage scaling, and can be inefficient at higher operating voltages. To overcome these shortcomings, switched capacitor (or equivalently switched inductor) converters utilize an array of capacitors for charge storage and a feedback circuit continuously switches between capacitors to maintain output voltages. Single-inductor multiple-ouput (SIMO) DC-DC converters further improve on switched capacitor designs by using a time-division multiplexing system to offer multiple output voltages while only requiring a single inductor for operation.

To estimate the area impact of voltage regulation, several SIMO designs were surveyed for various technology generations. Shown in Figure 74, area requirements range from $13.3mm^2$ in a $0.5\mu m$ process to $1mm^2$ in a $0.13\mu m$ process. The area-per-output plot shows a clear trend in area improvements with successive technology generations. The spread in per-output area requirements for $0.5\mu m$ technology designs does, however, indicate that design can have a significant impact on total area consumption. This is further evidenced by the fact that designs with roughly equivalent output counts (designs 3 and 6) have vastly different area requirements. From the survey, best and worst case area per voltage source is estimated at $0.36mm^2$ and $3.33mm^2$ respectively. This amounts to area requirements of

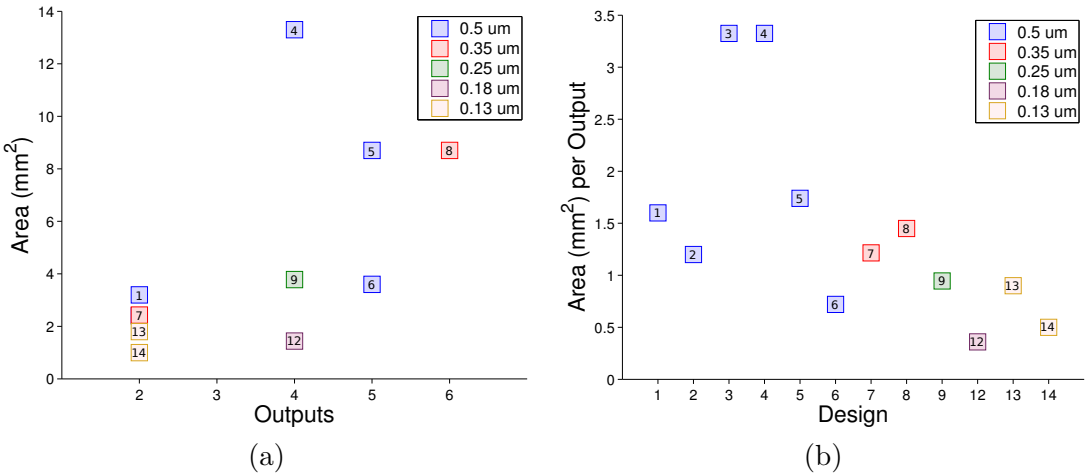


Figure 74: A comparison of various single-inductor multiple-output DC-DC voltage converter designs across five technology generations: $0.5\mu m$, $0.35\mu m$, $0.25\mu m$, $0.18\mu m$, $0.13\mu m$. (a) Design area versus voltage outputs. (b) Area per output for each design. Design clearly determines area requirements per output, however, technology dictates overall area consumption.

1: Woo 2-Output Buck/Boost [96]
2: Ma 2-Output Boost [49, 50]
3: Belloni 4-Output Boost [4]
4: Belloni 4-Output Buck [6]
5: Le 5-Output Boost [43]
6: Seol 5-Output Buck/Boost [73]
7: Belloni 2-Output Buck [6]
8: Lee 6-Output Buck [44]
9: Huang 4-Output Buck/Boost [31]
10: Bondade 3-Output Buck/Boost [10]
11: Belloni 4-Output Buck [5]
12: Parayandeh 4-Output Buck [64]
13: Belloni 2-Output Buck [6]
14: Zhang 2-Output Boost [102]
15: Bondade 2-Output Buck [9]

$0.72mm^2$ in the best case and $6.66mm^2$ in the worst to provide 2 distinct voltages levels.

To address the voltage regulator losses, the same 15 SIMO voltage converters were compared for efficiency (Figure 75). As BIVOS requires step-up voltage conversion, best and worse case Buck converters are considered: Belloni's 2-output converter [6] and Bondade's 2-output converter [9]. Based on these two designs, SIMO efficiency is estimated between 87% and 74%, best and worst case respectively.

5.4 H.264 Video Decoding Software

An open-source software decoder written by Martin Fiedler was chosen as the evaluation platform for H.264 video decoding [24]. The decoder was written in C and implements a minimal H.264 decoding solution. It was modified to incorporate the previously outlined C++ simulator to act as a PCMOS emulator. Under this configuration, a six-tap, 9-bit, FIR filter was implemented using the C++ simulator. The FIR filter design was then compiled to a library and integrated with the H.264 video decoding software (Figure 76).

When decoding videos, the decoder software first initializes the FIR filter library for

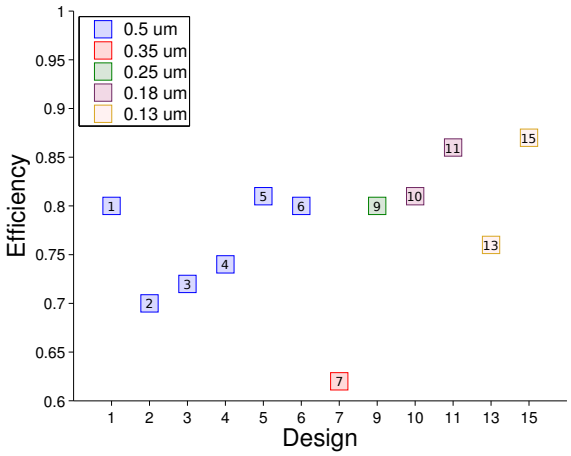


Figure 75: A comparison of the minimum reported voltage conversion efficiency for 15 single-inductor multiple-output DC-DC voltage converter designs across 5 technology generations: $0.5\mu m$, $0.35\mu m$, $0.25\mu m$, $0.18\mu m$, $0.13\mu m$. Technology largely drives voltage-conversion efficiency, however, designs 7 and 13 indicate that circuit design does play a role in efficiency.

1: Woo 2-Output Buck/Boost [96]
2: Ma 2-Output Boost [49, 50]
3: Belloni 4-Output Boost [4]
4: Belloni 4-Output Buck [6]
5: Le 5-Output Boost [43]
6: Seol 5-Output Buck/Boost [73]
7: Belloni 2-Output Buck [6]
8: Lee 6-Output Buck [44]
9: Huang 4-Output Buck/Boost [31]
10: Bondade 3-Output Buck/Boost [10]
11: Belloni 4-Output Buck [5]
12: Parayandeh 4-Output Buck [64]
13: Belloni 2-Output Buck [6]
14: Zhang 2-Output Boost [102]
15: Bondade 2-Output Buck [9]

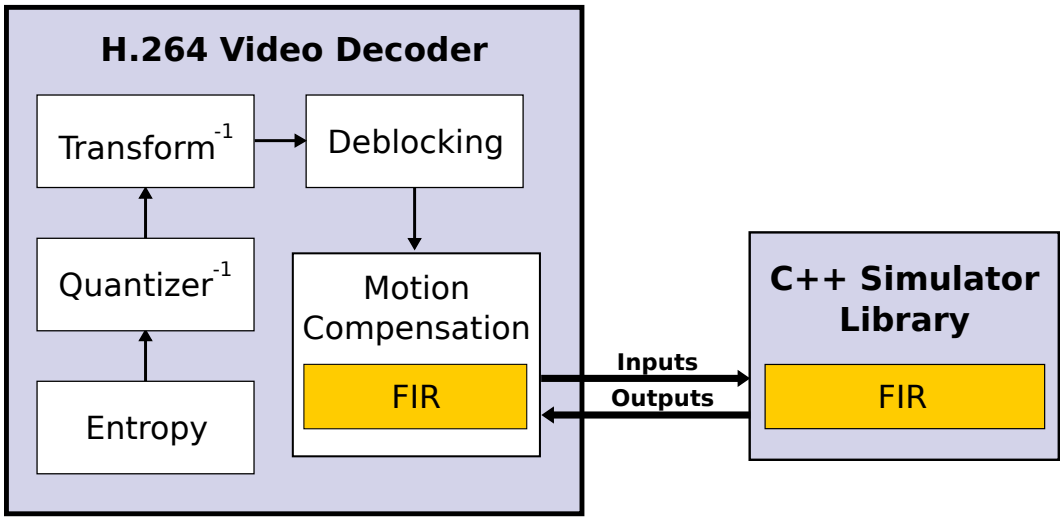


Figure 76: H.264 simulator data flow where the FIR filter within the motion compensation stage of the video decoder has been replaced with a PCMOS emulator. Data intended for the FIR filter is offloaded to a pre-compiled PCMOS FIR filter library that simulates probabilistic behavior. FIR filter outputs are then returned to the video decoder to continue standard processing.

Table 11: FIR Layout Implementations

Design	**Width** (μm)	**Height** (μm)	**Area** (mm^2)	**Penalty** (%)
Nominal	506.3	292.3	0.147	–
BIVOS	515.4	294.7	0.151	2.6%

specific bias parameters. As part of the motion estimation algorithm, normal FIR filter processing is bypassed and FIR filter inputs are delivered to the C++ simulator library. The library processes the inputs, injecting errors as determined by the biasing parameters, and returns the resulting output. Once the FIR filter output is received by the decoding software, it continues normal operation to generate the resulting frame. Once decoding is complete, ideal and test frames are compared to determine SNR.

5.5 Video Decoding Results

Outlined in Table 11, the resulting nominal FIR filter design required $0.147mm^2$ silicon area with a height of $292.3\mu m$ and a width of $506.3\mu m$. By comparison, the BIVOS design required $0.151mm^2$ with a height of $294.7\mu m$ and a width of $515.4\mu m$. The primary area expense for the BIVOS design was the necessary inclusion of five inverter/level-converter pairs for voltage conversion at each of the array multipliers (the inverter/level-converter pairs fit within dead space in the standard adder).

Beyond the area required for layout, the BIVOS design requires additional area for DC-DC converters needed to generate the two distinct voltage levels. Using the area per-output data from the 15 SIMO designs surveyed in Section 5.3, best and worst-case area requirements for voltage conversion are estimated at $0.72mm^2$ and $6.66mm^2$ respectively. When compared to the $0.151mm^2$ required for the BIVOS FIR filter implementation, it is clear that voltage regulation can render BIVOS solutions impractical from designs with tight area constraints. To some extent this regulation cost can be offset by employing the same regulators over multiple components operating on a single PCMOS co-processor. With several units utilizing the same biasing configuration the combined area could potentially approach that required for voltage conversion. Still, DC-DC converters represent a substantial area

Table 12: FIR Filter Energy Consumption and SNR

Design	**Energy** ($pJ/clock$)	**Reduction** (%)	**PSNR** (dB @ 1f)	**PSNR** (dB @ 18f)	**PSNR** (dB @ 36f)
Nominal	18.6	-	-	-	-
Reduced	15.3	26.1	37.2	20.1	14.3
BIVOS ($1.4V$)	14.7	29.1	50.0	40.6	38.1
BIVOS ($1.2V$)	13.7	33.9	45.8	37.2	35.0

requirement in BIVOS designs even when voltage binning is employed to place limits on the number of voltages required.

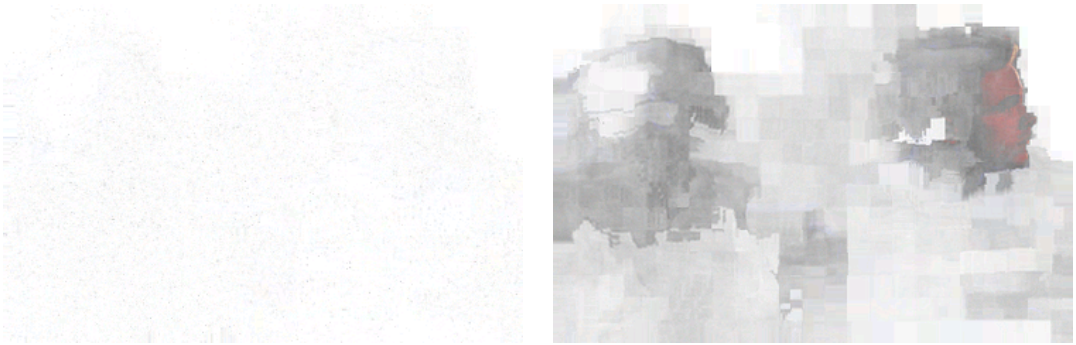
Simulated using the previously outlined H.264 decoder software, configurations were tested for nominal CMOS, reduced-precision, and BIVOS operation. Data normalization was employed for both reduced-precision and BIVOS operation to mitigate the impact of any bit errors. This was accomplished by left-shifting each coefficient three bit positions to fully utilize the available bit width and right-shifting results by three bit positions to realign data. As each inter (predicted) frame is based on a previously rendered frame, any errors that occurred were accumulated across multiple frames resulting in the highest error rates at the end of an inter-frame sequence. To indicate the impact of each solution at various sizes of inter-frame sequences, peak-signal-to-noise ratio (PSNR) was compared at frames 1, 18, and 36 of a single sequence.

Energy consumption and PSNR for each of the configurations tested is summarized in Table 12. BIVOS configurations were simulated over a range of operating points to highlight the opportunity to trade accuracy for energy savings based on application requirements. In all cases, BIVOS designs provide higher energy savings with a better PSNR than a reduced-precision solution. A BIVOS solution optimized for image quality (biased at $1.4V$) allows a 29.1% reduction in energy consumption with a PSNR of $38.1dB$. Optimized for energy consumption (biased at $1.2V$), a BIVOS solution yields 33.9% energy savings with a PSNR of $35.0dB$. By comparison, an optimal reduced-precision solution at 11 bits results in a 26.1% energy reduction with a PSNR of $14.3dB$.

Frames comparing the "low-energy" BIVOS and reduced-precision implementations are

(a) (b)

Figure 77: Identical frames from the movie X-Men 2 as decoded using (a) BIVOS and (b) reduced-precision implementations. The BIVOS solution achieved an energy reduction of 33.9% at a PSNR of $38.1dB$ compared to a 26.1% energy reduction at $14.3dB$ for the reduced-precision solution.



(a) (b)

Figure 78: Noise introduced into identical frames from the movie X-Men 2 as decoded using (a) BIVOS and (b) reduced-precision implementations. The BIVOS solution achieved an energy reduction of 33.9% at a PSNR of $38.1dB$ compared to a 26.1% energy reduction at $14.3dB$ for the reduced-precision solution.

shown in Figures 77 and 78. Where the reduced-precision solution severely degrades image quality, the BIVOS solution only slightly alters pixel intensity. At low resolution, as depicted here, the impact of BIVOS is largely imperceptible. Full resolution images for each configuration tested are shown in Appendix F.

Based on the bounds established in Section 5.3 for voltage-regulator losses, energy consumption with voltage-converter efficiency is summarized in Table 13. From Table 13, it is evident that regulation efficiency is a critical factor in the effectiveness of a BIVOS solution. Assuming minimal regulator efficiency, neither BIVOS implementation (with energy savings of only 4.2% and 10.6% for the "high-quality"and "low-power" biases respectively) is

Table 13: Efficiency Comparison

Design	**Energy** ($pJ/clock$)	**Efficiency** (%)	**Total Energy** ($pJ/clock$)	**Reduction** (%)
Nominal	18.6	90	20.7	-
Reduced	15.3	90	17.0	17.8
BIVOS ($1.4V$)	14.7	74	19.8	04.2
		87	16.8	18.5
BIVOS ($1.2V$)	13.7	74	18.5	10.6
		87	15.7	24.0

able to compete with the 17.8% energy savings realized through reduced-precision CMOS. Assuming regulator efficiency on par with that used for a standard design, however, yields BIVOS solutions that surpass a reduced-precision solution with energy savings at 18.5% and 24.0% (again, for the "high-quality"and "low-power" biases respectively). As outlined in Table 12, this is accomplished while substantially improving on SNR when compared to a reduced-precision design.

Shown here for H.264 video decoding, BIVOS solutions outperform an *energy-equivalent*, reduced-precision solution given the appropriate design conditions. The primary drawback to a implementing a BIVOS solution is the additional requirements imposed by voltage regulation. From Table 13, it is obvious that regulator efficiency is critical to the effectiveness of a BIVOS implementation. If regulator efficiency is not on par with single voltage designs, any savings realized through BIVOS can easily be overwhelmed. Further, the area penalty incurred for regulation of multiple voltage planes can quickly eliminate BIVOS for designs with little area to spare. For designs that can meet the voltage regulation requirements, however, a BIVOS solution can deliver substantial energy reductions with minimal impact to application quality.

CHAPTER VI

CONCLUSIONS AND POTENTIAL DIRECTIONS

6.1 Contributions: Biased Voltage Overscaling

For applications that inherently require probability, probabilistic computing is an obvious fit. When realized through PCMOS, solutions offer a high quality source of randomness that substantially reduces computational complexity compared to more conventional approaches. This translates to PCMOS designs that reduce the consumption of both silicon area and power. Applied to probabilistic applications, the result can be enormous power savings over more conventional designs.

It is much less obvious how probabilistic computing applies to applications that are inherently deterministic. These applications were designed with the assumption that the underlying hardware operates in a deterministic fashion. Errors are not anticipated and, as a general rule, the expectation is that errors should not be permitted. While previous works allowed for computational errors, any that did so corrected the resulting errors under the assumption that deterministic operation was a necessity.

The primary contribution of this work is the extension of probabilistic computing to these inherently deterministic applications. Specifically, the novel biasing technique presented here represents a highly unorthodox solution that makes application resiliency possible. By biasing error generation to low-order bit positions, error magnitude is limited such that the resulting computational approximations are tolerable at the application level.

When realized through PCMOS, BIVOS solutions are capable of significantly reducing the power consumption of CMOS devices. While these solutions are unable to match the extreme power savings PCMOS offers for probabilistic applications, BIVOS provides power savings beyond what is possible with standard implementations. The inability to realize the potential of probabilistic applications stems primarily from the massive hardware reductions that are possible by implementing probabilistic algorithms in PCMOS. Where probabilistic

applications are able to capitalize on probability to reduce hardware complexity, resilient applications must operate in spite of it.

Beyond the hardware benefits enjoyed by probabilistic applications, there are a variety of other factors further limiting the overall effectiveness of a BIVOS solution. Application accuracy requirements ultimately determine what degree of voltage overscaling is possible. These requirements place an upper limit on the number of bits that can be biased within a circuit, beyond which biasing is impossible due to the error magnitude introduced through probabilistic operation. Independent of the accuracy requirements, the act of biasing places a lower limit on the number of bits that can be biased within a circuit by creating propagation waves that increase power consumption at unbiased, higher-order bit positions. Combined with circuit structure, the resulting upper and lower biasing bounds allow a limited operational range for BIVOS solutions. Once this operating range is coupled with the added overhead of voltage generation and conversion, BIVOS design can be challenging.

Despite the difficulties imposed by BIVOS design, BIVOS solutions perform well when compared to other error-prone techniques. Reduced-precision designs achieve equivalent energy savings, however, the technique is incapable of matching the accuracy delivered by BIVOS. Similarly, standard (uniform) voltage overscaling is also capable of decreasing power consumption, although the accuracy sacrifice is well beyond that required by BIVOS designs. As a result, BIVOS offers the potential for resilient applications to minimize sacrifices in application quality while maximizing energy savings.

While the work presented here has shown the applicability of BIVOS to a specific computing technology with a specific noise source, namely CMOS and thermal noise, the applicability of the technique is far broader. Independent of the underling noise source, or computing technology, the work here has shown that it is possible to perform useful computation in a non-deterministic fashion. The biasing technique shows how accuracy requirements can be relaxed in a way that maintains application performance. In a similar fashion, biasing can be applied to other noise sources and technologies to realize energy savings through probabilistic computing.

6.2 Investigating Delay as a Source of Noise

Propagation delay, in particular, represents a "noise" source that could potentially be well suited for PCMOS operation. In conventional, synchronous logic a circuit is sampled at fixed intervals to determine signal states. Switching signal states takes time at each CMOS transistor and signals can only be sampled after CMOS transistors have had ample time to process any input signals. Because not all paths through a circuit are equivalent, the time required for changing input signals to propagate to circuit outputs is variable. As a result, the circuit sampling rate, or clock period, is determined to accommodate worst case propagation delays.

In practice, however, the worst case propagation paths are rarely activated. Instead, propagation delay through the circuit will vary with changing input patterns. In a fashion similar to reducing the noise margin for thermal noise, increasing the circuit sampling rate will reduce the margin for propagation delay. Shown in Figure 79, if a circuit is sampled after a changing input signal has had time to propagate through the entire circuit, the associated output signals are correct. If the circuit has not had time to propagate changing signals through the entire circuit, however, the associated output signals are incorrect resulting in a bit error.

Over the course of many samples propagation delay through the circuit will vary. The resulting frequency of these varying propagation delays then forms a probability density function (PDF), not unlike that formed by thermal noise. For a given clock period the probability of an error $(1 - p)$, and conversely the probability of correctness (p), can then be determined from the PDF.

Operating as a pseudo-noise source, propagation delay then presents two opportunities for gains through PCMOS operation. First, for a fixed, nominal voltage distribution, a circuit's clock rate can be increased beyond that permitted by deterministic operation allowing for increased performance without a power penalty. Conversely, a circuit's clock rate can be maintained while the voltage distribution is reduced allowing for decreased energy consumption without a performance penalty. Combining the two, PCMOS operation can potentially yield increased performance with reduced energy consumption.

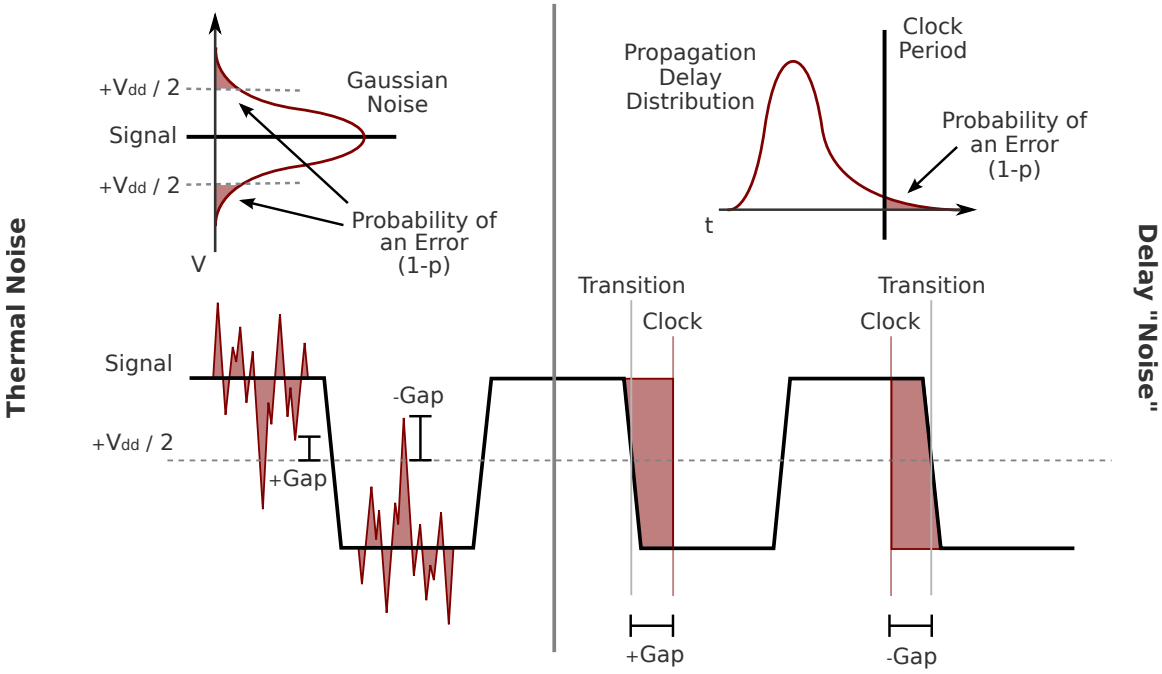


Figure 79: Treated as a pseudo-noise source, propagation delay shows several similarities to thermal noise. By over clocking a circuit, the frequency of the resulting propagation delays forms a probability-density function that can then be used to estimate the probability of an error based on clock period.

6.3 Implications for Optical Computing

Beyond CMOS, probabilistic computing and biasing holds promise as a more general power saving technique. Optical computing is one emerging technology were BIVOS could stand to make a substantial impact. The technology transmits data using light, or photons, rather than electric current. Because light paths will not interfere with each other, crossing signals can simply be overlapped reducing interconnect overhead. More importantly, the use of photons for data transmission has the advantage of resistance-free transmission channels. Where electrical wires suffer from electrical resistance resulting in induced heat, optical channels virtually eliminate waste heat generation [97].

Due to these resistance free-transmission channels, optical computing should consume less power than an electronic equivalent. In practice, however, this is hardly the case. Shot noise, a result of fluctuations in the number of detectable photons, quickly overcomes optical signals requiring a boost in signal strength and an associated increase in power consumption. Since the shot noise in an optical channel is greater than the thermal noise in

an electrical channel, optical channels require more power to overcome the underlying noise sources. Over long distances the transmission losses incurred through electrical resistance will outweigh any shot losses incurred by an optical channel, however, at the lengths required to span a silicon die an optical device will end up consuming more power than an electronic equivalent.

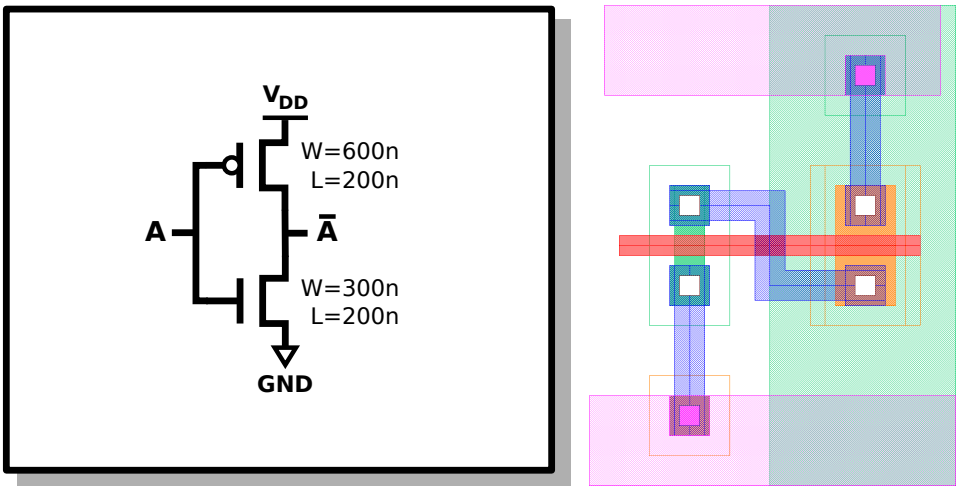
In many ways, shot noise is already affecting optical computing in a fashion similar to that projected of thermal noise and electronic computing. Both are additive noise sources and both are statistically random processes—shot noise follows a Poisson distribution as opposed to the Gaussian distribution observed by thermal noise. More importantly, shot noise places lower limits on the energy consumption of optical systems requiring a baseline of transmission power to maintain noise margins. This behavior is identical to the projected impact of thermal noise on voltage scaling in future CMOS generations.

While optical computing is far from mainstream, BIVOS represents a solution that could benefit the technology immediately. Chances are that other, yet to be discovered, computing techniques and technologies could benefit from the application of biased computing in a similar fashion. As a technique, biasing offers a solution to perform useful computations using approximations rather than absolutes. It is this departure from the more conventional mindset of deterministic computation that represents the innovation of this work.

Reducing the power consumption of computing systems is a theme that spans the history of electronic computation. From the earliest vacuum-tube machines to the modern processors in use today, there has been a persistent need for improvements in energy efficiency. Given the power wall that the computing industry is facing and projections that this wall will continue to be a barrier to computing, it is likely that there will be a need for energy-efficient computation well into the future.

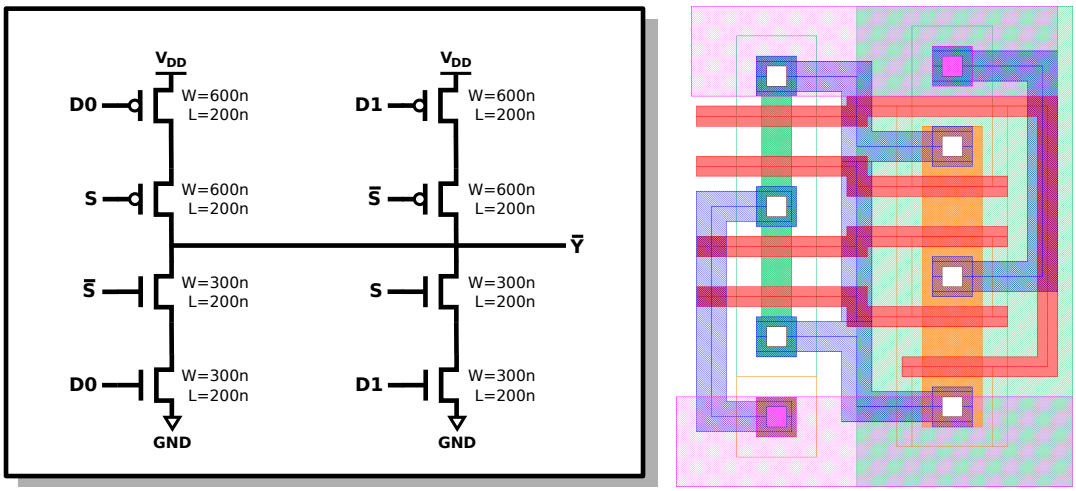
APPENDIX A

STANDARD CELL IMPLEMENTATIONS



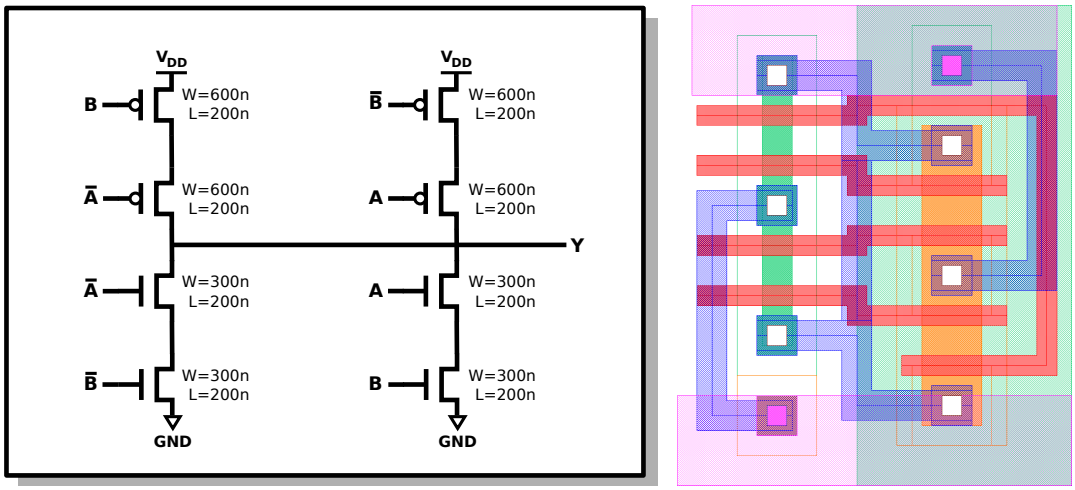
(a) (b)

Figure 80: Inverter implementation: (a) transistor schematic with transistor sizing for minimal, balanced input capacitance and (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $3.65\mu m$ and an area of $17.52\mu m^2$.



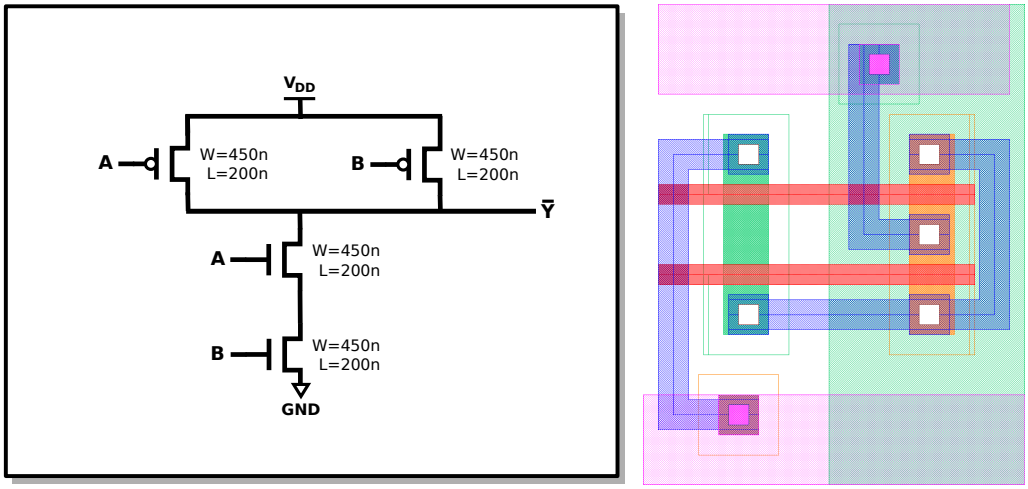
(a) (b)

Figure 81: Multiplexor implementation: (a) transistor schematic with transistor sizing for minimal, balanced input capacitanceand (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $3.95\mu m$ and an area of $18.96\mu m^2$.



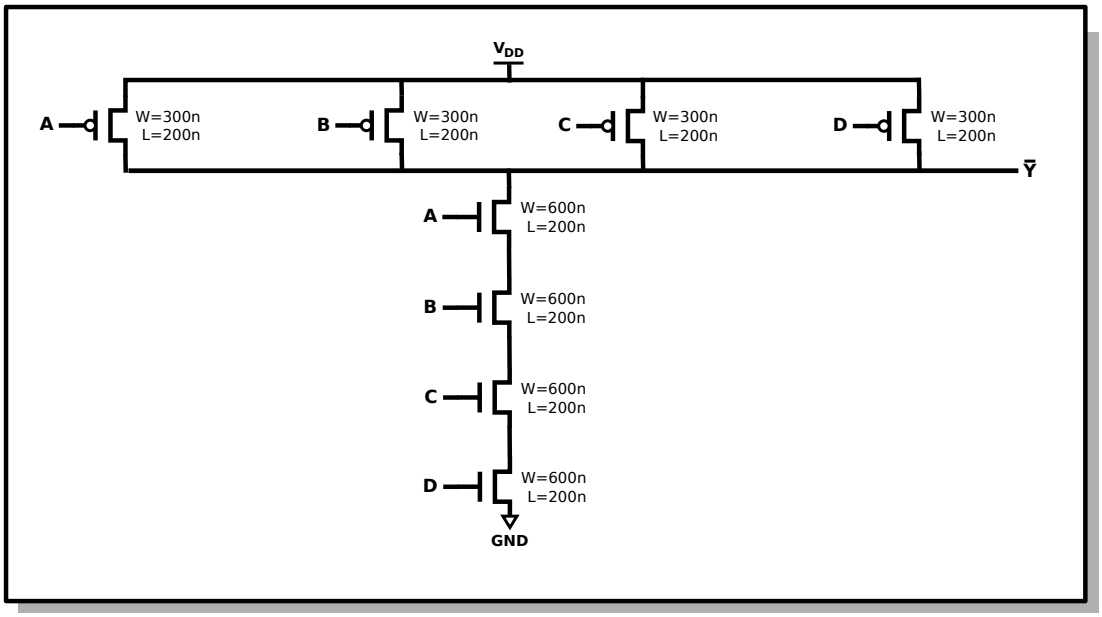
(a) (b)

Figure 82: Exclusive-or implementation: (a) transistor schematic with transistor sizing for minimal, balanced input capacitanceand (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $3.95\mu m$ and an area of $18.96\mu m^2$.

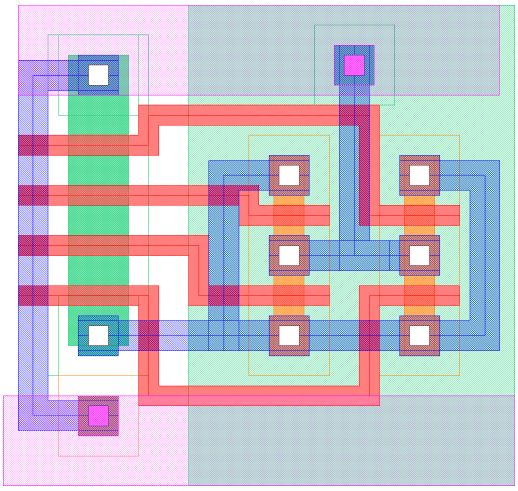


(a) (b)

Figure 83: Nand implementation: (a) transistor schematic with transistor sizing for minimal, balanced input capacitanceand (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $3.80\mu m$ and an area of $18.24\mu m^2$.

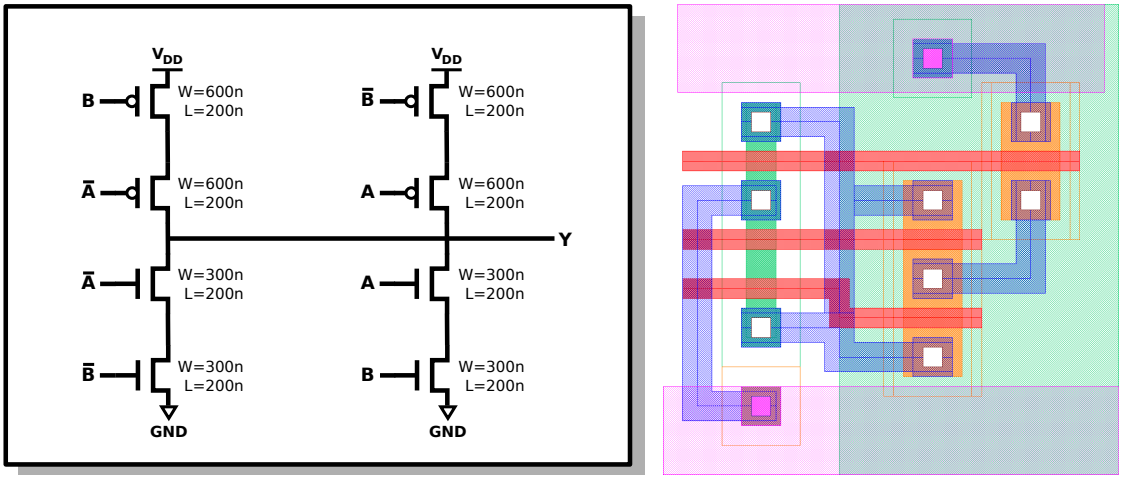


(a)



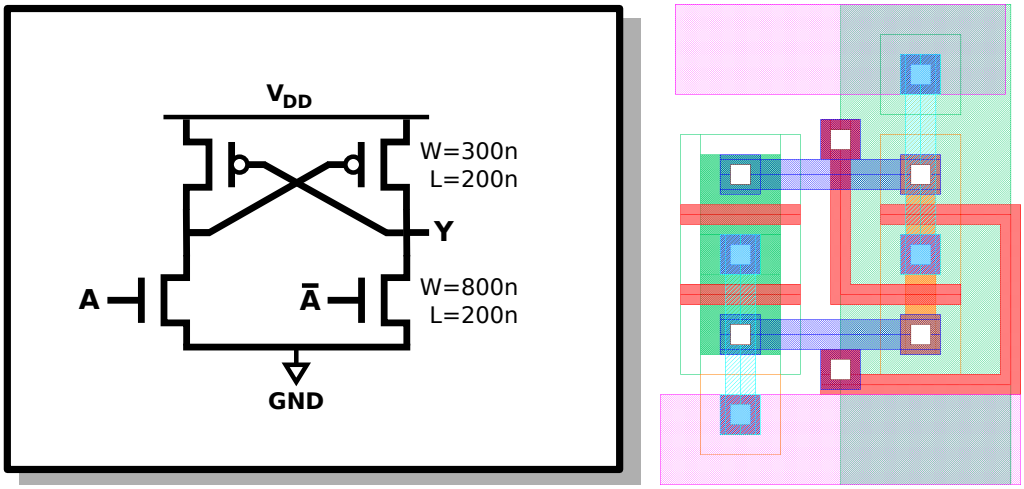
(b)

Figure 84: Four-input NAND implementation: (a) transistor schematic with transistor sizing for minimal, balanced input capacitanceand (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $5.10\mu m$ and an area of $24.48\mu m^2$.



(a) (b)

Figure 85: And-or-invert implementation: (a) transistor schematic with transistor sizing for minimal, balanced input capacitanceand (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $4.65\mu m$ and an area of $22.32\mu m^2$.



(a) (b)

Figure 86: Level converter implementation: (a) transistor schematic with transistor sizing for minimal power consumptionand (b) VLSI layout with cell pitch set $4.8\mu m$ resulting in a width of $3.60\mu m$ and an area of $17.52\mu m^2$.

APPENDIX B

CMOS RIPPLE-CARRY ADDER

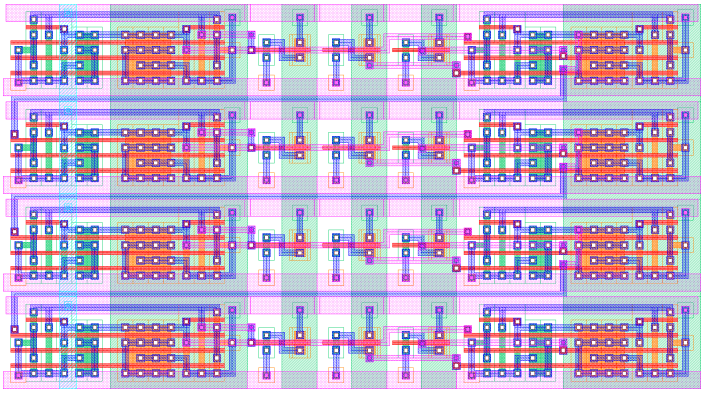


Figure 87: Standard CMOS implementation for an 8-bit, ripple-carry adder. Full adders are divided into four, two-bit rows to create a roughly square design at 36.6μm by 20.1μm and a total area of 735.66μm^2.

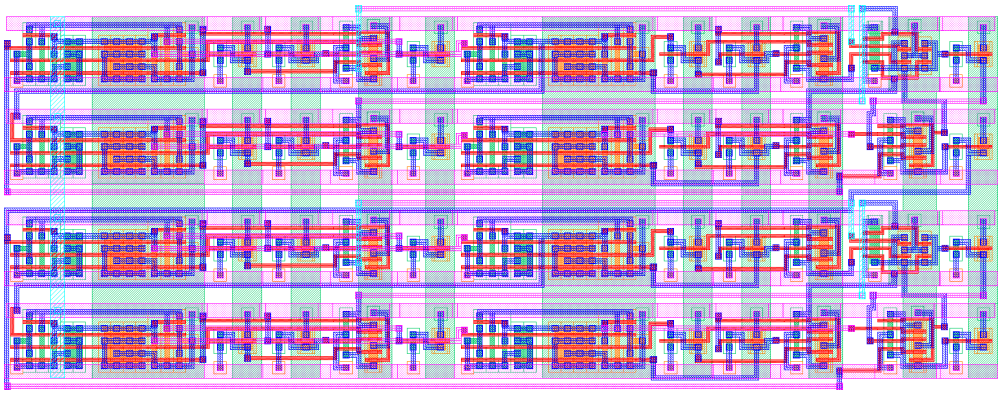


Figure 88: Standard CMOS implementation for an 8-bit, block-propagate adder. Full adders are divided into four, two-bit rows with propagate logic inserted between resulting in a rectangular design at 63.5μm by 24.4μm and a total area of 1549.40μm^2.

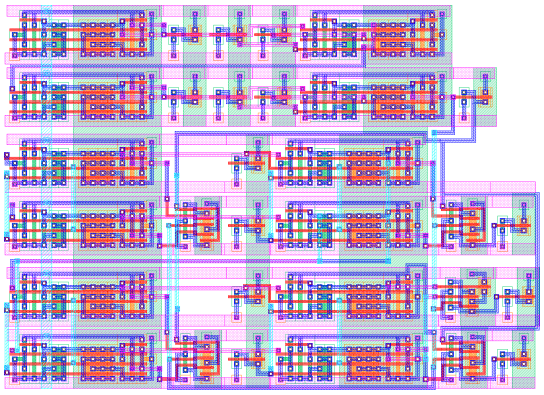


Figure 89: Standard CMOS implementation for an 8-bit carry-select adder. Full adders are divided into six, two-bit rows with the replicated select logic residing on rows three, four, five, and six. The resulting design is roughly square at 43.9μm by 31.4μm and a total area of 1378.46μm^2.

APPENDIX C

INVERTER BIVOS RIPPLE-CARRY ADDER

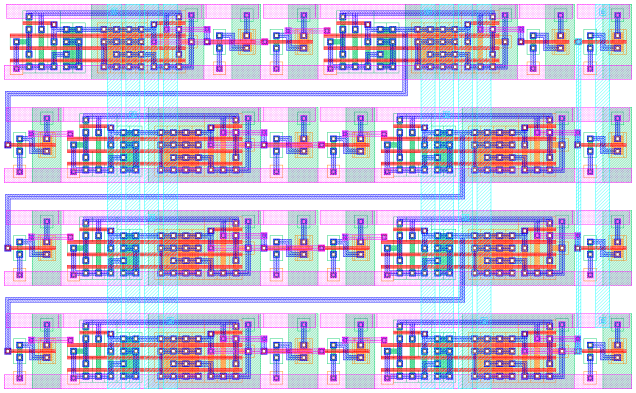


Figure 90: BIVOS implementation for an 8-bit, ripple-carry adder. Full adders are divided into four, two-bit rows to create a rectangular design at $40.2\mu m$ by $24.6\mu m$ and a total area of $988.92\mu m^2$.

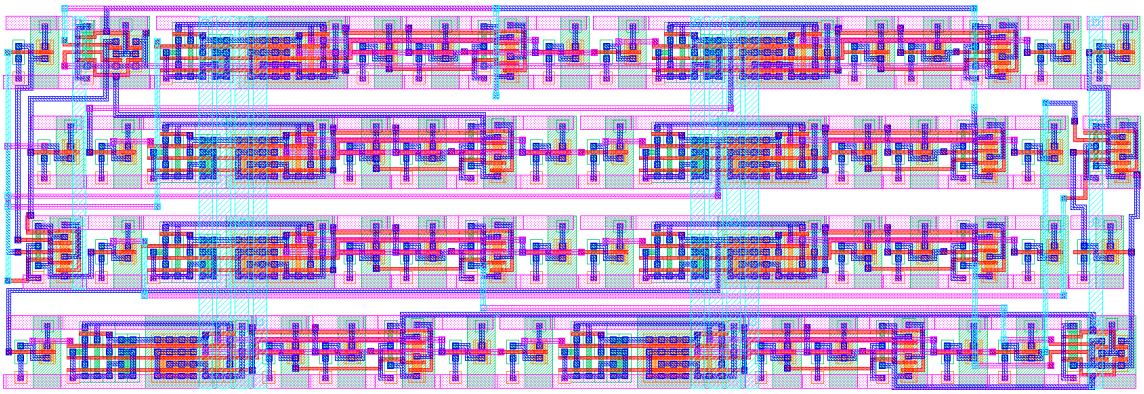


Figure 91: BIVOS implementation for an 8-bit, block-propagate adder. Full adders are divided into four, two-bit rows with propagate logic inserted between resulting in an elongated design at $75.4\mu m$ by $25.3\mu m$ and a total area of $1907.62\mu m^2$.

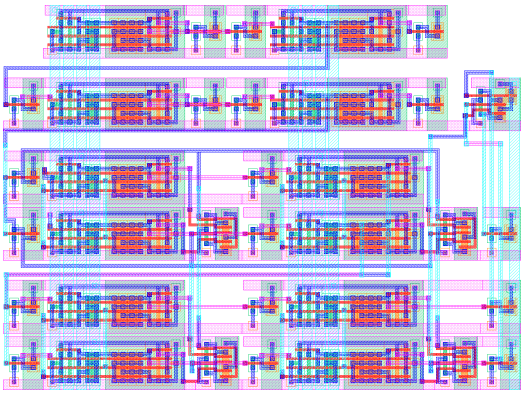


Figure 92: BIVOS implementation for an 8-bit, carry-select adder. Full adders are divided into six, two-bit rows with the additional logic replicated for the select portion on the lower four row. The resulting design is roughly square at $46.9\mu m$ by $34.8\mu m$ and a total area of $1632.12\mu m^2$.

APPENDIX D

LEVEL-CONVETER BIVOS RIPPLE-CARRY ADDER

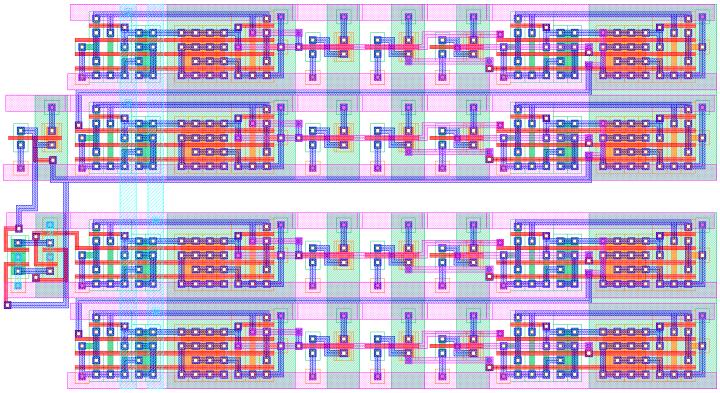


Figure 93: BIVOS implementation for an 8-bit, ripple-carry adder utilizing traditional level conversion. Full adders are divided into four, two-bit rows to create a rectangular design at $40.2\mu m$ by $21.6\mu m$ and a total area of $868.32\mu m^2$.

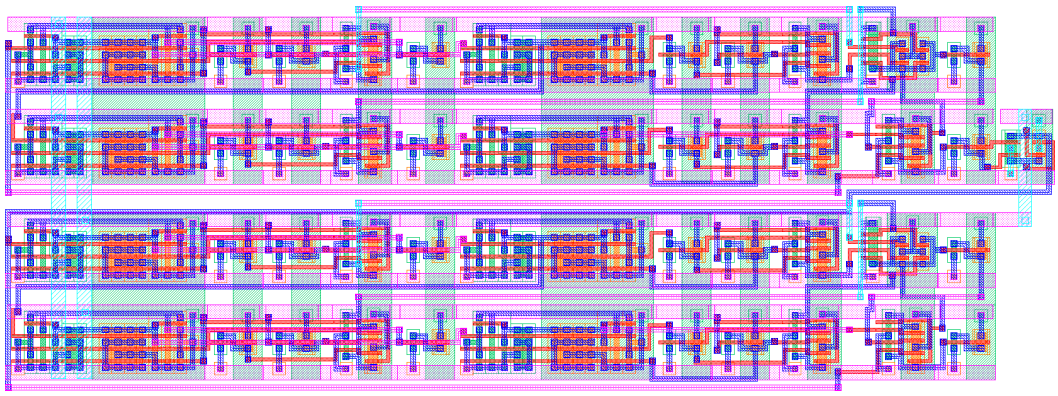


Figure 94: BIVOS implementation for an 8-bit, block-propagate adder utilizing traditional level conversion. Full adders are divided into four, two-bit rows with propagate logic inserted between resulting in an elongated design at $67.3\mu m$ by $24.6\mu m$ and a total area of $1655.58\mu m^2$.

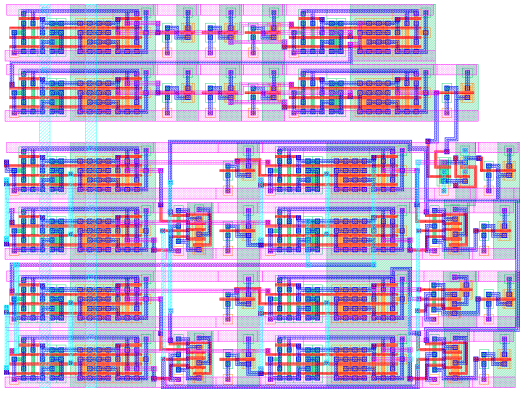


Figure 95: BIVOS implementation for an 8-bit, carry-select adder utilizing traditional level conversion. Full adders are divided into six, two-bit rows with the additional logic replicated for the select portion on the lower four row. The resulting design is roughly square at $43.9\mu m$ by $32.6\mu m$ and a total area of $1431.14\mu m^2$.

APPENDIX E

H.264 VIDEO DECODING FRAMES

Figure 96: H.264 video decoding of the movie X-Men 2 using 18-bit, standard CMOS for frame 1 in a sequence of 36 inter (prediction) frames.

Figure 97: H.264 video decoding of the movie X-Men 2 using 18-bit, standard CMOS for frame 18 in a sequence of 36 inter (prediction) frames.

Figure 98: H.264 video decoding of the movie X-Men 2 using 18-bit, standard CMOS for frame 36 in a sequence of 36 inter (prediction) frames.

Figure 99: H.264 video decoding of the movie X-Men 2 using 11-bit, reduced-precision CMOS for frame 1 in a sequence of 36 inter (prediction) frames. At frame one, blocking is already present around the actor on the right.

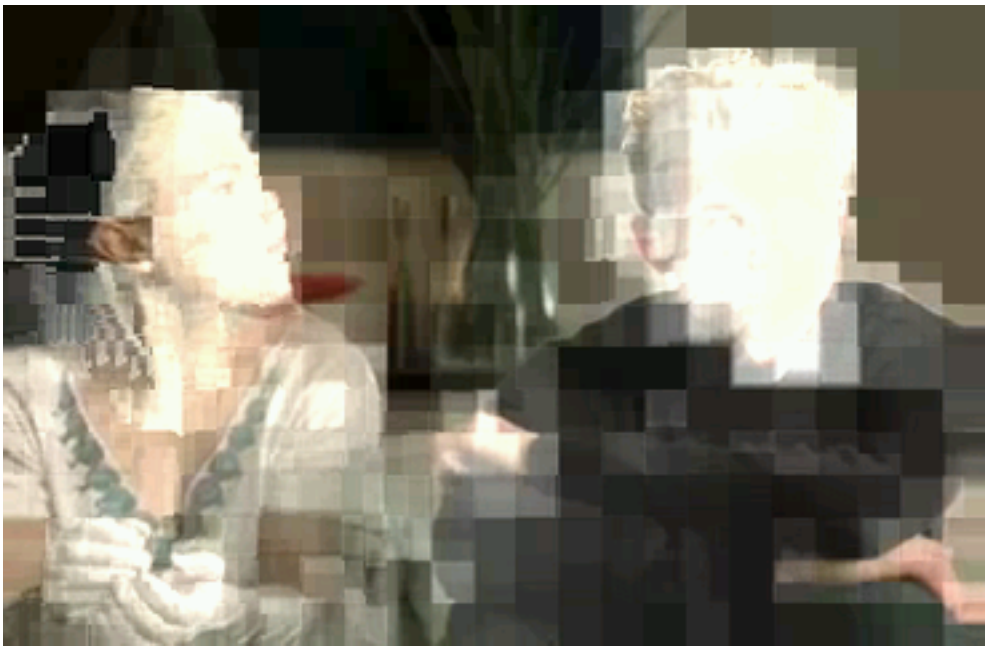


Figure 100: H.264 video decoding of the movie X-Men 2 using 11-bit, reduced-precision CMOS for frame 18 in a sequence of 36 inter (prediction) frames. By frame 18, both actors faces are fully obscured due to noise introduced through reduced-precision operation.

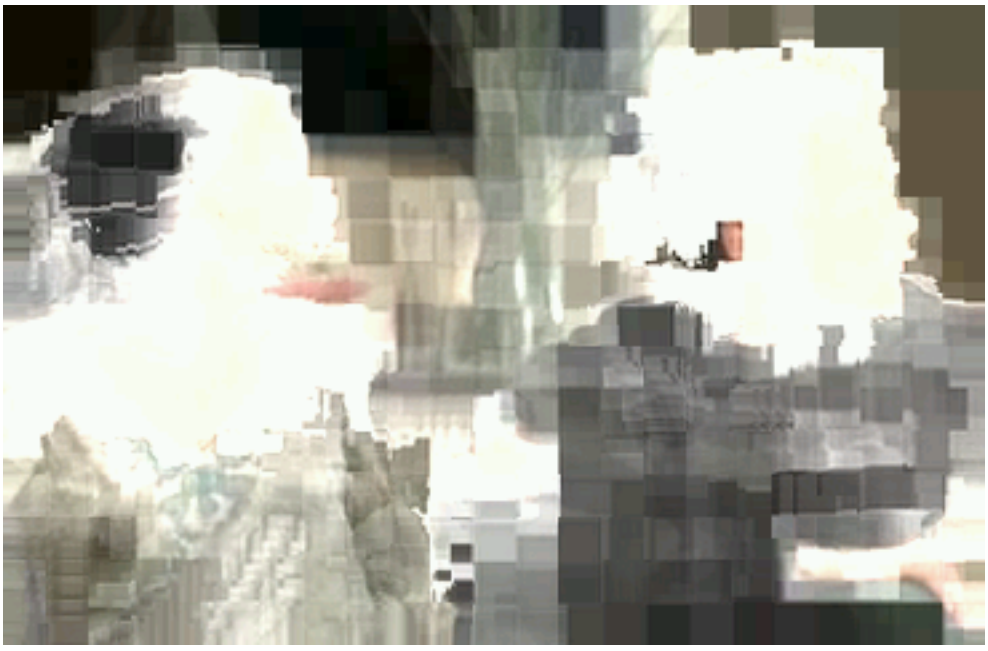


Figure 101: H.264 video decoding of the movie X-Men 2 using 11-bit, reduced-precision CMOS for frame 36 in a sequence of 36 inter (prediction) frames. At frame 36, a large portion of the scene is obscured due to noise introduced through reduced-precision operation.

Figure 102: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.4V$ for frame 1 in a sequence of 36 inter (prediction) frames. At frame one, no discernable noise is evident due to BIVOS operation.

Figure 103: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.4V$ for frame 18 in a sequence of 36 inter (prediction) frames. At frame 18, still no clear evidence of noise is present.

Figure 104: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.4V$ for frame 36 in a sequence of 36 inter (prediction) frames. Even at frame 18, it is difficult to detect the noise introduced through BIVOS operation.

Figure 105: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.2V$ for frame 1 in a sequence of 36 inter (prediction) frames. No discernable noise is evident due to BIVOS operation.

Figure 106: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.2V$ for frame 18 in a sequence of 36 inter (prediction) frames. Slight pixelation is evident in the actor's faces at full resolution due to BIVOS operation.

Figure 107: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.2V$ for frame 36 in a sequence of 36 inter (prediction) frames. As in frame 18, slight pixelation is evident in the actor's faces at full resolution due to BIVOS operation.

APPENDIX F

H.264 VIDEO DECODING NOISE

Figure 108: H.264 video decoding of the movie X-Men 2 using 11-bit, reduced-precision CMOS for frame 1 in a sequence of 36 inter (prediction) frames. Noise is barely discernable at frame one.

Figure 109: H.264 video decoding of the movie X-Men 2 using 11-bit, reduced-precision CMOS for frame 18 in a sequence of 36 inter (prediction) frames. By frame 18 the noise has accumulated significantly.

Figure 110: H.264 video decoding of the movie X-Men 2 using 11-bit, reduced-precision CMOS for frame 36 in a sequence of 36 inter (prediction) frames. At frame 36, the noise becomes strong throughout the entire frame.

Figure 111: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.4V$ for frame 1 in a sequence of 36 inter (prediction) frames. As was the case in the actual frame, noise is difficult to detect.

Figure 112: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.4V$ for frame 18 in a sequence of 36 inter (prediction) frames. Again, noise is difficult to detect.

Figure 113: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.4V$ for frame 36 in a sequence of 36 inter (prediction) frames. Even at frame 36, noise is still difficult to detect.

Figure 114: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.2V$ for frame 1 in a sequence of 36 inter (prediction) frames. Similar to the higher voltage solution, noise is difficult to detect at frame one.

Figure 115: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.2V$ for frame 18 in a sequence of 36 inter (prediction) frames. By frame 18, noise is slightly discernable.

Figure 116: H.264 video decoding of the movie X-Men 2 using 18-bit, BIVOS biased to $1.2V$ for frame 36 in a sequence of 36 inter (prediction) frames. More pronounced than frame 18, noise at frame 36 is still only slightly detectable.

REFERENCES

[1] Apple, "Apple Computers: Electrical Specifications." Available at: `http://support.apple.com/kb/TA37853?viewlocale=en_US`, Sept. 2010.

[2] Apple, "Macbook air technical specifications." Available at: `http://www.apple.com/macbookair/specs.html`, Oct. 2010.

[3] Austin, T., Blaauw, D., Mudge, T., and Flautner, K., "Making typical silicon matter with razor," *Computer*, vol. 37, no. 3, pp. 57–65, 2004.

[4] Belloni, M., Bonizzoni, E., Kiseliovas, E., Malcovati, P., Maloberti, F., Peltola, T., and Teppo, T., "A 4-output single-inductor DC-DC buck converter with self-boosted switch drivers and 1.2A total output current," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pp. 444 –626, Feb. 2008.

[5] Belloni, M., Bonizzoni, E., and Maloberti, F., "On the design of single-inductor multiple-output DC-DC buck converters," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 3049 –3052, May 2008.

[6] Belloni, M., Bonizzoni, E., and Maloberti, F., "On the design of single-inductor multiple-output DC-DC buck converters," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 3049 –3052, May 2008.

[7] Bhanu, A., Lau, M. S. K., Ling, K.-V., Mooney III, V. J., and Singh, A., "A more precise model of noise based pcmos errors," in *Proceedings of the 2010 Fifth IEEE International Symposium on Electronic Design, Test & Applications*, DELTA '10, (Washington, DC, USA), pp. 99–102, IEEE Computer Society, 2010.

[8] Boliolo, A., Benini, L., de Micheli, G., and Ricco, B., "Gate-level power and current simulation of CMOS integrated circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, pp. 473 –488, Dec. 1997.

[9] Bondade, R. and Ma, D., "A DLL-regulated SIMO power converter for DVS-enabled power-aware VLSI systems," in *Circuits and Systems, 2009. MWSCAS '09. 52nd IEEE International Midwest Symposium on*, pp. 961 –964, Aug. 2009.

[10] Bondade, R. and Ma, D., "Hardware-software co-design of an embedded power management module with adaptive on-chip power processing schemes," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 617 –620, May 2010.

[11] Burr, J. and Peterson, A., "Ultra low power CMOS technology," in *Idaho Univ., The 1991 3rd NASA Symposium on VLSI Design 12 p (SEE N94-18337 04-33)*, 1991.

[12] Calhoun, B. H. and Chandrakasan, A., "Characterizing and modeling minimum energy operation for subthreshold circuits," in *ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design*, pp. 90–95, 2004.

[13] Calhoun, B. H., Daly, D. C., Verma, N., Wentzloff, D. D., Wang, A., Cho, S.-H., and Chandrakasan, A. P., "Design considerations for ultra-low energy wireless microsensor nodes," *IEEE Transactions on Computers*, vol. 54, no. 6, pp. 727–740, 2005.

[14] Carpenter, B. E., "Observed relationships between size measures of the Internet," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 5–12, 2009.

[15] Carpenter, B. E., "Observed relationships between size measures of the Internet or is the Internet really just a star network after all?." Available at: `http://apricot.vip.net.id/www.apricot2010.net/__data/assets/pdf_file/00%07/18844/APOPS-Plenary-I_01_BGP-Growth-over-15-years_Brian-Carpenter.pdf`, 2009.

[16] Chakrapani, L., Akgul, B. E. S., Cheemalavagu, S., Korkmaz, P., Palem, K., and Seshasayee, B., "Ultra-efficient (embedded) SOC architectures based on probabilistic CMOS (PCMOS) technology," *Proc. of Design Automation and Test in Europe (DATE)*, Mar. 2006.

[17] Chandrakasan, A. P. and Brodersen, R. W., *Low power digital CMOS design*. Norwell, MA: Kluwer Academic Publishers, 1995.

[18] Chang, J. and Pedram, M., "Energy minimization using multiple supply voltages," *Proc. of IEEE Transactions on VLSI Systems*, vol. 5, pp. 436 – 443, Dec. 1997.

[19] Cheemalavagu, S., Korkmaz, P., and Palem, K. V., "Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship," in *Proc. of The 2004 International Conference on Solid State Devices and Materials*, (Tokyo, Japan), pp. 402 – 403, Sept. 2004.

[20] Cheemalavagu, S., Korkmaz, P., Palem, K. V., Akgul, B. E. S., and Chakrapani, L. N., "A probabilistic CMOS switch and its realization by exploiting noise," in *Proc. of In IFIP-VLSI SoC*, (Perth, Western Australia), Oct. 2005.

[21] Chong, W., Hariyama, M., and Kameyama, M., "Low-power field-programmable vlsi using multiple supply voltages," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, pp. 3298–3305, Dec. 2005.

[22] Cox, P. M., Betts, R. A., Jones, C. D., Spall, S. A., and Totterdell, I. J., "Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model," *Nature*, vol. 408, pp. 184–187, Nov. 2000.

[23] Cremer, C., Eichhammer, W., Friedewald, M., Georgieff, P., Rieth-Hoerst, S., Schlomann, B., and Zoche, P., "Energy consumption of information and communication technology (ICT) in Germany up to 2010." Available at: `http://publica.fraunhofer.de/eprints/urn:nbn:de:0011-n-223629.pdf`, Jan. 2003.

[24] Fiedler, M., "H.264 decoder; c-source code." Available at: `http://keyj.s2000.ws/?page_id=41`, May 2008.

[25] Friedman, E., "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, pp. 665 –692, May 2001.

[26] George, B., Yeap, G., Wloka, M., Tyler, S., and Gossain, D., "Power analysis for semi-custom design," in *Custom Integrated Circuits Conference, 1994., Proceedings of the IEEE 1994*, pp. 249 –252, May 1994.

[27] George, J., Marr, B., Akgul, B. E. S., and Palem, K. V., "Probabilistic arithmetic and energy efficient embedded signal processing," in *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, (New York, NY, USA), pp. 158–168, ACM, 2006.

[28] Gwennap, L., "What is a microprocessor?." Available at: `http://www.mdronline.com/editorial/edit24_39.html`, Sept. 2010.

[29] Hanchate, N. and Ranganathan, N., "A new technique for leakage reduction in CMOS circuits using self-controlled stacked transistors," in *VLSID '04: Proceedings of the 17th International Conference on VLSI Design*, (Washington, DC, USA), p. 228, IEEE Computer Society, 2004.

[30] Hedge, R. and Shanbhag, N. R., "Soft digital signal processing," *IEEE Transactions on VLSI*, vol. 9, pp. 813 – 823, Dec. 2001.

[31] Huang, M.-H. and Chen, K.-H., "Single-inductor multi-output (SIMO) DC-DC converters with high light-load efficiency and minimized cross-regulation for portable devices," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 1099 –1111, Apr. 2009.

[32] IBM, "7090 Data Processing System." Available at: `http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP7090.html`, Sept. 2010.

[33] Johnson, M. C., Somasekhar, D., Chiou, L.-Y., and Roy, K., "Leakage control with efficient use of transistor stacks in single threshold CMOS," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 10, no. 1, pp. 1–5, 2002.

[34] Kalva, H., "The H.264 video coding standard," *Multimedia, IEEE*, vol. 13, pp. 86 –90, Oct. 2006.

[35] Kim, N., Austin, T., Blaauw, D., Mudge, T., Flautner, K., Hu, J., Irwin, M., Kandemir, M., and Vijaykrishnan, N., "Leakage current - Moore's law meets static power," *IEEE Computer*, vol. 36, pp. 68–75, Dec. 2003.

[36] Kish, L. B., "End of Moore's law: Thermal (noise) death of integration in micro and nano electronics," *Physics Letters A*, vol. 305, pp. 144–149, Dec. 2002.

[37] Kogge, P., Bergman, K., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hill, K., and et al., "Exascale computing study: Technology challenges in achieving exascale systems." DARPA Information Processing Techniques Office, Jan. 2008.

[38] Kohler, A. and Erdmann, L., "Expected environmental impacts of pervasive computing," *Human and Ecological Risk Assessment*, vol. 10, pp. 831 – 852, Oct. 2004.

[39] Koomey, J., "Estimating total power consumption by servers in the U.S. and the world." Available at: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.5562&rep=re%p1&type=pdf`, Feb. 2007.

[40] Korkmaz, P., Akgul, B. E. S., Palem, K. V., and Chakrapani, L. N., "Advocating noise as an agent for ultra-low energy computing: Probabilistic CMOS devices and their characteristics," *Japanese Journal of Applied Physics, SSDM Special Issue Part 1*, pp. 3307–3316, Apr. 2006.

[41] Kulkarni, S. H. and Sylvester, D., "Power distribution techniques for dual Vdd circuits," in *Proc. of the 2006 conference on Asia South Pacific design automation*, (Yokohama, Japan), pp. 838 – 843, 2006.

[42] Landauer, R., "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, pp. 183 –191, July 1961.

[43] Le, H.-P., Chae, C.-S., Lee, K.-C., Cho, G.-H., Wang, S.-W., Cho, G.-H., and il Kim, S., "A single-inductor switching DC-DC converter with 5 outputs and ordered power-distributive control," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pp. 534 –620, Feb. 2007.

[44] Lee, K.-C., Chae, C.-S., Cho, G.-H., and Cho, G.-H., "A PLL-based high-stability single-inductor 6-channel output DC-DC buck converter," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pp. 200 –201, Feb. 2010.

[45] M. Lau, K. V. Ling, A. B. and III, V. J. M., "Error rate prediction for probabilistic circuits with more general structures," in *The 16th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI2010)*, pp. 220–225, 2010.

[46] M. Lau, K. V. Ling, Y. C. C. and Bhanu, A., "A general mathematical model of probabilistic ripple-carry adders," in *Design Automation and Test in Europe (DATE)*, Mar. 2010.

[47] M. Lau, K. V. Ling, Y. C. C. and Bhanu, A., "Modeling and optimization of probabilistic ripple-carry adders," in *Proceedings of the 2010 Fifth IEEE International Symposium on Electronic Design, Test & Applications*, pp. 201–206, Jan. 2010.

[48] Ma, D. and Bondade, R., "Enabling power-efficient DVFS operations on silicon," *Circuits and Systems Magazine, IEEE*, vol. 10, no. 1, pp. 14 –30, 2010.

[49] Ma, D., Ki, W.-H., Tsui, C.-Y., and Mok, P., "A 1.8 V single-inductor dual-output switching converter for power reduction techniques," in *VLSI Circuits, 2001. Digest of Technical Papers. 2001 Symposium on*, pp. 137 –140, 2001.

[50] Ma, D., Ki, W.-H., Tsui, C.-Y., and Mok, P., "Single-inductor multiple-output switching converters with time-multiplexing control in discontinuous conduction mode," *Solid-State Circuits, IEEE Journal of*, vol. 38, pp. 89 – 100, Jan. 2003.

[51] Manzak, A. and Chaktrabarti, C., "Variable voltage task scheduling algorithms for minimizing energy/power," *Proc. of IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, pp. 270 – 276, Apr. 2003.

[52] Meindl, J. and Davis, J., "The fundamental limit on binary switching energy for terascale integration (TSI)," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 1515 –1516, Oct. 2000.

[53] Min, K., Kawaguchi, H., and Sakurai, T., "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: An alternative to clock-gating scheme in leakage dominant era," in *Proc. of the 28th European Solid-State Circuits Conference*, pp. 679 – 682, Sept. 2002.

[54] Mitra, S. K., *Digital Signal Processing A Computer Based Approach*. McGraw-Hill, third ed., 2006.

[55] Mudge, T., "Power: A first-class architectural design constraint," *IEEE Computer*, vol. 34, pp. 52–58, Apr. 2001.

[56] Mutho, S., Douseki, T., Matsuya, Y., Aoki, T., Shigematsu, S., and Yamada, J., "1-V power suppy high-speed digital circuit technology with multithreshold-voltage cmos," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 847–854, Aug. 1995.

[57] Narendra, S., Borkar, S., De, V., Antoniadis, D., and Chandrakasan, A., "Scaling of stack effect and its application for leakage reduction," in *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*, (New York, NY, USA), pp. 195–200, ACM, 2001.

[58] Natori, K. and Sano, N., "Scaling limit of digital circuits due to thermal noise," *Journal of Applied Physics*, vol. 83, pp. 5019–5024, May 1998.

[59] Padgett, W. T. and Anderson, D. V., *Fixed-point Signal Processing*. Synthesis Lectures on Signal Processing, Morgan & Claypool Publishers, 2009.

[60] Palem, K. V., "Proof as experiment: Probabilistic algorithms from a thermodynamic perspective," in *Proc. Intl. Symposium on Verification (Theory and Practice)*, (Taormina, Sicily), June 2003.

[61] Palem, K. V., "Proof as experiment: Probabilistic algorithms from a thermodynamic perspective," in *Proc. Intl. Symposium on Verification (Theory and Practice)*, (Taormina, Sicily), June 2003.

[62] Palem, K. V., "Energy aware computing through probabilistic switching: A study of limits," *IEEE Trans. Computer*, vol. 54, no. 9, pp. 1123–1137, 2005.

[63] Palem, K. V., Chakrapani, L. N., Akgul, B. E. S., and Korkmaz, P., "Realizing ultra low-energy application specific SoC architectures through novel probabilistic CMOS (PCMOS) technology," in *Proc. of the International Conference on Solid State Devices and Materials*, (Tokyo, Japan), pp. 678 – 679, Sept. 2005.

[64] Parayandeh, A., Stupar, A., and Prodic, A., "Programmable digital controller for multi-output DC-DC converters with a time-shared inductor," in *Power Electronics Specialists Conference, 2006. PESC '06. 37th IEEE*, pp. 1 – 6, June 2006.

[65] Park, J. C. and Mooney, V. J., "Sleepy stack leakage reduction," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 11, pp. 1250–1263, 2006.

[66] Park, J. C., Mooney, V. J., and Srinivasan, S. K., "Combining data remapping and voltage/frequency scaling of second level memory for energy reduction in embedded systems," *Microelectronics Journal*, vol. 34, no. 11, pp. 1019–1024, 2003.

[67] Pratt, V., "Anatomy of the pentium bug," in *TAPSOFT '95: Theory and Practice of Software Development, volume 915 of Lecture Notes in Computer Science*, pp. 97–107, Springer-Verlag, 1995.

[68] Puri, R., Stok, L., Cohn, J., Kung, D., Pan, D., Sylvester, D., Srivastava, A., and Kulkarni, S., "Pushing ASIC performance in a power envelope," in *Proc. of the 40th Design Automation Conference (DAC'03)*, p. 788, 2003.

[69] Puttaswamy, K., Choi, K.-W., Park, J. C., Mooney, V. J., Chatterjee, A., and Ellervee, P., "System level power-performance trade-offs in embedded systems using voltage and frequency scaling of off-chip buses and memory," in *Proceedings of the 15th International Symposium of Systems Synthesis*, (Kyoto, Japan), pp. 225–230, 2002.

[70] Roth, K. W. and McKenney, K., "Energy consumption by consumer electronics in U.S. residences." Available at: `http://www.ce.org/AboutCEA/CEAInitiatives/3638.asp`, Jan. 2007.

[71] Sabolc Pal, Zeljko Lukac, M. D., "H.264 decoder," in *XIII Telekomunikacioni forum TELFOR*, (Beograd, Serbia), Nov. 2005.

[72] Sarin, H. and McNelly, A., "A power modelling and characterization method for logic simulation," in *Custom Integrated Circuits Conference, 1995., Proceedings of the IEEE 1995*, pp. 363 –366, May 1995.

[73] Seol, K.-S., Woo, Y.-J., Cho, G.-H., Cho, G.-H., Lee, J.-W., and il Kim, S., "Multiple-output step-up/down switching DC-DC converter with vestigial current control," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pp. 442 –443, Feb. 2009.

[74] Sharangpani, H. and Barton., M., "Statistical analysis of floating point flaws in the pentium processor." Available at: `http://www.intel.com/support/processors/pentium/sb/cs-013005.htm`, 1994.

[75] Shepard, K. L., "Conquering noise in deep-submicron digital ICs," *IEEE Design and Test of Computers*, vol. 15, pp. 51 – 62, Jan. - Mar. 1998.

[76] Stein, K.-U., "Noise-induced error rate as limiting factory for energy per operation in digital ICs," *Solid-State Circuits, IEEE Journal of*, vol. 12, pp. 527 – 530, Oct. 1977.

[77] The World Bank, "Energy use (kg of oil equivalent per capita)." Available at: `http://data.worldbank.org/topic/energy-and-mining`, Jan. 2011.

[78] U.S. Department of Energy, "International energy outlook 2007." Available at: `http://www.eia.doe.gov/oiaf/ieo/index.html`, May 2007.

[79] U.S. Department of Energy, "American recovery & reinvestment act." Available at: `http://www1.eere.energy.gov/recovery/`, Jan. 2011.

[80] U.S. Energy Information Administration, "Frequently Asked Questions - Electricity." Available at: `http://www.eia.doe.gov/ask/electricity_faqs.asp#electricity_use_home`, Sept. 2010.

[81] U.S. Energy Information Administration, "Annual energy review 2009." Available at: http://www.eia.doe.gov/emeu/aer/consump.html, Jan. 2011.

[82] Usami, K. and Horowitz, M., "Clustered voltage scaling technique for low-power design," in *ISLPED '95: Proceedings of the 1995 international symposium on Low power design*, pp. 3–8, 1995.

[83] Vitousek, P. M., "Beyond global warming: Ecology and global change," *Ecology*, vol. 75, pp. 1861–1876, Oct. 1994.

[84] Von Neumann, J. and Burks, A. W., *Theory of self-reproducing automata*. Urbana, IL: University of Illinois Press, 1966.

[85] Wang, A. and Chandrakasan, A., "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 310–319, Jan. 2005.

[86] Weste, N. H. E. and Harris, D., *CMOS VLSI Design: A Circuits and Systems Perspective, 3/E*. Addison-Wesley, 2005.

[87] Wiegand, T., Sullivan, G., Bjontegaard, G., and Luthra, A., "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, pp. 560 –576, July 2003.

[88] Wikipedia, "Colossus computer." Available at: http://en.wikipedia.org/wiki/Colossus_computer, Sept. 2010.

[89] Wikipedia, "ENIAC." Available at: http://en.wikipedia.org/wiki/ENIAC, Sept. 2010.

[90] Wikipedia, "Grid Compass." Available at: http://en.wikipedia.org/wiki/Grid_Compass, Sept. 2010.

[91] Wikipedia, "History of computing hardware." Available at: http://en.wikipedia.org/wiki/History_of_computing_hardware#Colossus, Sept. 2010.

[92] Wikipedia, "IBM 7090." Available at: http://en.wikipedia.org/wiki/IBM_7090, Sept. 2010.

[93] Wikipedia, "MOBIDIC." Available at: http://en.wikipedia.org/wiki/MOBIDIC, Sept. 2010.

[94] Wikipedia, "Pentium FDIV bug." Available at: http://en.wikipedia.org/wiki/Pentium_FDIV_bug, Oct. 2010.

[95] Wikipedia, "Vacuum tube." Available at: http://en.wikipedia.org/wiki/Vacuum_tube, Sept. 2010.

[96] Woo, Y.-J., Le, H.-P., Cho, G.-H., Cho, G.-H., and Kim, S.-I., "Load-independent control of switching DC-DC converters with freewheeling current feedback," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pp. 446 –626, Feb. 2008.

[97] Woods, D. and Naughton, T. J., "Optical computing," *Applied Mathematics and Computation (Special issue on Physics and Computation)*, vol. 215, pp. 1417–1430, Oct. 2009.

[98] Yajnanarayana, V., Subramaniyan, R., and Schuette, M., "Techniques to improve motion compensation performance of H264 video decoder using a vector processor," in *Communications and Information Technologies, 2007. ISCIT '07. International Symposium on*, pp. 1082 –1087, Oct. 2007.

[99] Yeap, G. K., *Practical low power digital VLSI design*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[100] Yeh, Y. and Kuo, S., "An optimization-based low-power voltage scaling technique using multiple supply voltages," in *Proc. of IEEE Internaitonal Symposium on ISCAS 2001*, vol. 5, pp. 535 – 538, May 2001.

[101] Yeh, Y., Kuo, S., and Jou, J., "Converter-free multiple-voltage scaling techniques for low-power CMOS digital design," *Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 172 – 176, Jan. 2001.

[102] Zhang, Y. and Ma, D., "Digitally controlled integrated pseudo-CCM SIMO converter with adaptive freewheel current modulation," in *Applied Power Electronics Conference and Exposition (APEC), 2010 Twenty-Fifth Annual IEEE*, pp. 284 –288, Feb. 2010.